CWDS
Child Welfare Digital Services

Child Welfare Digital Services Project

Release Management Plan

January 2019

Revision History

Revision / Version #	Date of Release	Author	Summary of Changes
V 1.0	5/30/2017	John Simko	Original Document
V 1.1	6/29/2017	John Simko	More awesome updates
V 1.2	7/18/2017	John Simko	Changes based on feedback
V 1.3	9/7/2017	John Simko	Updates based on C&B feedback
V 1.4	10/6/2017	John Simko	Updates based on C&B feedback and other changes to make it up to date
V 1.5	5/15/2018	John Simko	Back and better than ever
V 1.6	1/31/2019	John Simko	Updated the entire document

Table of Contents

1 Introduction

1.1 Purpose

This document describes the Release Management Plan (hereinafter referred to as the "plan") for the Child Welfare Services – California Automated Response and Engagement System (CWS-CARES) Project (hereinafter referred to as the "Project"). The purpose of this plan is to document how the Project will plan, build, test, deploy, manage, and govern CWS-CARES releases.

1.2 Scope

The scope of this plan covers all roles, activities and artifacts necessary to support the project's release management processes.

1.2.1 In-Scope

- Release Management
- Roadmap
- Continuous Delivery Pipeline
- Release Types
- Release Principles
- Release Rollout
- Release-level Definition of Done (aka Release Readiness Checklist)
- Release Notes
- Release Governance

1.2.2 Out-of-Scope

- Agile Practices, including digital service team ceremonies (e.g., daily stand-up, sprint reviews, sprint planning, scrum of scrums, Sprint and Definition of Done)
- Implementation Team Activities
    - Rollout approach (covered in the Statewide Implementation Plan)
    - Approach for determining State, county, and tribal Go-Live Readiness (covered in the Statewide Implementation Plan)
    - Training
    - OCM
    - Implementation
- DevOps activities

1.3 Risks

Title	Description

Risk 1727 - Lack of overall product roadmap	Without knowledge of the shared vision for the product roadmap, the Digital Services teams may not be in concert on priorities and instead concentrate work on features locally important to their goals. Stakeholders are unaware of future release planning and are not able to proactively take steps to prepare staff and business processes for new functionality.

1.4 Integration with other CWDS Plans

This plan integrates with Change Management, Configuration Management and Test Management processes.

Quality Management Plan

The Quality Management Plan defines how quality will be managed throughout the project lifecycle of the Child Welfare Services-New System Project (hereafter call "Project").

Risk and Issue Management Plan

The Risk and Issue Management Plan includes the process, constraints and approach to be used by the CWS-CARES project to identify, analyze, plan, implement, monitor and close project risks and issues during the entire life of the project.

Change Management Plan

The Change Management Plan describes the process of managing changes to the Child Welfare Services-New System (CWS-CARES) project, artifacts, application code, deliverables at a strategic, tactical, and operational level. The purpose of this Change Management Plan is to establish a standardized agile change management approach for the approval and tracking of proposed changes for the CWS-CARES project.

Configuration Management Plan

The Configuration Management Plan describes how the Project will identify State service assets and configuration items (hereinafter referred to as "SA's", and "CI's") including internal deliverables and work products, and State owned software products. The Project will record all SA's and CI's, list their attributes and relationships, as well as protect them from unauthorized changes.

Test Management Approach

The CWDS Test Management Approach describes an overall framework and a set of principles, best practices, and guidelines for how testing will occur across all CWDS digital services. This document defines the overall testing strategy, but does not contain a complete set of detailed processes and procedures.

CWDS Implementation Plan

The CWDS Implementation Plan describes the State's overarching vision for implementing the various CWS-CARES Digital Services and primarily focuses on the State and Implementation Contractor's direct work with the staff in the Orgs who have been identified to participate in this implementation effort. This plan highlights the three main entities involved in implementation: the State implementation team, the implementation contractors, and the Orgs.

Statewide Implementation Plan

The CWS-CARES Statewide Implementation Plan describes the methodology, tools, and resources the Implementation Team uses to deliver CWDS implementation services to the 60 transitioning Organizations (Orgs).

1.5 Document Maintenance

The CWDS Release Management Plan will be updated as processes and procedures change. A minor version change does not change the intent of the document and consists of spelling, grammatical and minor corrections. A major version is when a document's content is changed and represents a change in intent, change in process, or procedures. Please refer to the CWDS Configuration Management for further detail on version control.

During development of this plan, the guidelines and standards provided through the Project's Quality Management Plan will apply; specifically all peer review requirements must be met.

2 Plan Approach

The CWDS release management approach is based on the Scaled Agile Framework® (SAFe) v4.0 for Lean Software and Systems Engineering approach of managing the development and delivery of software. This framework helps align the CWDS digital service teams to a common business context, vision, and program objectives using a cadence-based and fixed period of time.

One of the key components of SAFe® v4.0 that the project is following is the Agile Release Train (ART). Directly from the SAFe® website[1], "the ART aligns teams and helps manage risk and variability by providing cadence and synchronization." The project will follow SAFe®'s guidelines in organizing, planning, and executing its ART. Specifically, the project is electing to a single ART that can support a single system and organized by feature teams (i.e., Digital Services).

The ART aligns teams to a common mission via a comprehensive and integrated vision, roadmap, and product backlog. Development of the solution occurs on a standard cadence, but the teams can release at any time. In SAFe® terms, this is referred to as "Develop on a Cadence and Release Any Time". Regarding the development cadence,

[1] http://v4.scaledagileframework.com/agile-release-train/ (visited on 7/1/17)

SAFe® prescribes using a Program Increment (PI), which is a cadence-based interval for building and validating a full system increment, demonstrating value and getting fast feedback. For this project, a PI is comprised of six two-week iterations (i.e., sprints) of planned work to deliver solution value. Details on how the project will plan and manage the execution of the ART PIs can be found in Section 4 Program Increments. Details on how the project will manage releasing any time can be found in Section 5 Continuous Delivery (CD) Pipeline.

3 Roles and Responsibilities

The following table describe the roles and responsibilities of the CWS-CARES Project stakeholders in support of the ART activities.

Roles	Responsibilities
Release Manager	• Act as a servant leader and operate as a full-time "Chief Scrum Master" for the ART • Facilitates the Program Increment (PI) planning sessions • Schedules and facilitates the Scrum of Scrums meeting • Facilitates PI retrospectives • Manages the Continuous Delivery Pipeline • Maintain the Release Readiness Checklist • Consolidate Release Notes • Manages Releases
Agile Coach	• Guides the teams in the initial PI Planning process. • Trains the Release Manager in the latest Scaled Agile framework
Scrum Master(s)	• Leads their team in the process of preparing for release planning, attending and participating in the release planning activity, and facilitates the follow up work • Participates in the Scrum of Scrum meetings • Facilitates the PI retro
Development Teams	• Identifies backlog items they will likely need to realize the features. • Creates draft plans, visible to all, iteration by iteration. • Identifies risks and dependencies and drafts their initial team PI objectives. • Adds features to the program board at the point in time they will/can deliver the feature • Breaks the desired features into user stories needed to deliver (high level with an estimate, AC can be added later) • Completes any rework that arises out of the Management Review and Problem-Solving session of day 1. • Reviews the final desired features • Confirms feature dependencies • Finalizes draft plan, iteration by iteration. • Finalizes risks • Inputs remaining stories into Pivotal Tracker

Roles	Responsibilities
Implementation Teams	• Prepare Statewide Implementation Plan • Develop Rollout Schedule • Identify Core Counties for each Digital Service • Prepare Baseline Implementation Project and Org Schedules • Prepare Implementation • Kickoff Implementation in Org • Conduct OCM and Training • Facilitate Communication • Provide Implementation Oversight and Monitoring
Executive Leadership + Line of Business Owners	• As part of PI Planning: o Describes the future vision of the business and presents a perspective on how well current solutions are addressing current customer needs (this focuses on the comparison between the problem area and the plans for this release). o Describes the current state of the business and presents a perspective on how well current solutions are addressing the customer's needs. o Coordinate and approve the CWDS portfolio roadmap • The Executive Leadership Team (ELT) participates in the Release Governance Process • Participate in the project-level Risk and Issues resolution process
Service Manager(s)	• As part of PI Planning: o Highlights any changes from the previous PI planning meeting as well as any upcoming milestones that have been identified. o Participates in the management review and problem-solving sessions o Describes any changes to scope and/or resources that came out of the management review and problem-solving session. o Communicates planned features to their stakeholders • Works with their digital service team and core county users to identify the specific set of features and capabilities that would comprise of a Minimal Viable Product (MVP). • Coordinates with the Release Manager in moving release candidates through the Continuous Delivery Pipeline.
Product Owner(s)	• As part of PI Planning: o Presents the current program vision – typically represented by the next top 10 upcoming features (or whatever is being released for the release). o Presents their draft objectives and by what sprint they will deliver key items • Coordinates with the Release Manager in moving release candidates through the Continuous Delivery Pipeline. • Develop Release Notes
Solution Architect	• Defines the architecture runway that supports feature development • Presents the architecture vision and practices during PI Planning. • Participates in the Release Governance Process
Development Lead/Manager(s)	• Presents agile-supportive changes to development practices such as test automation and continuous integration during PI Planning.

Roles	Responsibilities
DevOps Engineering	• Presents the DevOps Engineering vision and practices during PI Planning. • Coordinates with the Release Manager in setting up and maintaining environments and pipeline infrastructure • Maintains the CD configuration management
Information Security Officer	• Presents security vision and practices during PI Planning • Verifies that a release candidate passes all CWDS Security Tests as part of the Continuous Delivery Pipeline
Checks and Balances	• Provide independent oversight of testing issues and areas of non-conformance in accordance to CA-PMF and IEEE (where appropriate). • Observe sprint reviews, release planning and execution and testing activities.
CWDS Quality Assurance	• Provide enterprise quality assurance support services to the state during the agile life cycle. • Ensure compliance with the Project Management Body of Knowledge (PMBOK) guidelines, California Project Management Methodology (CA-PMM), OSI Best Practices (http://www.bestpractices.osi.ca.gov/), industry standards, Information Technology Infrastructure Library (ITIL) standards. • Develop and maintain compliance to the CWDS Quality Management Plan. • Develop and maintain compliance to the CWDS Quality Metrics Plan. • Define the standards to which quality will be measured. • Measure and improve process and product quality.
QA Engineer(s)	• Build and maintain sets of more advanced (e.g., negative, edge case) feature tests, integration tests, regression tests and load/performance tests. • Work closely with Digital Service, Tech Platform, and DevOps teams through the entire project lifecycle to ensure overall software system quality is maintained. • Work across digital service teams to ensure integration across all services. • Work closely with Legacy Testing team to develop test scripts. • Gather/analyze requirements and develop test plans on new/existing features. • Do manual testing in the early phases of product development if automated tests are not ready. • Create, modify, execute and maintain feature, integration, and load/performance test scripts along with full regression tests. • Build and maintain a library of reusable scripts and processes. • Provide feedback to digital service, Tech Platform, and DevOps teams on software usability and functionality • Develop strong working knowledge of child welfare services business practices. • Log and track software defects. • Become a key source of institutional knowledge of how the system as a whole works.

4 Program Increments

4.1 Roadmap

The project's roadmap is used to communicate a high-level timeline of when core business processes as well as new or enhanced functionality is planned on being delivered. It is used to While currently under development, the project's roadmap will represent the business processes and new or enhance functionality that has been implemented, in progress (current PI), near term (next 3 PIs), and long term. The near term business processes and functionality is used to help plan and prioritize features for the PI Planning session.

4.2 PI Planning

The ART is initiated with a PI Planning session that occurs prior to the start of each PI and is facilitated by the Release Train Engineer (hereinafter referred to as the "Release Manager"). It typically takes place over one and a half days and includes all digital service team members that are participating in the development and delivery of CWS-CARES solution during the upcoming PI time period. It starts off with presentations of the business context and vision of the overall project and individual digital service teams that are participating in PI. This is followed by presentations of architecture vision and any agile-supportive changes to development practices such as test automation. DevOps engineering may present continuous integration and continuous delivery activities that are being advanced in the upcoming PI. Lastly, Information Security may present specific information security items that apply to the upcoming PI.

These presentations are then followed by breakouts wherein the teams are expected to accomplish the following:

- Estimate their capacity for the next six sprints
- Calculate their velocity in accordance with their capacity
- Identify their PI objectives / features. The objectives should follow the S.M.A.R.T (Specific, Measurable, Achievable, Realistic, Time-Bound) objective guidelines
- Decompose their desired features into high-level user stories
- Create a draft plan and add the features to the program board at the point in time they will/can deliver the feature. Features are only placed within the first five iterations (Iteration 6 is dedicated to Innovation and Planning activities)
- Identify risks using red sticky notes on the program board in the iterations when the risk will most likely be triggered
- Identify cross-team dependencies

The draft plan is represented on a program board, which highlights the new features, anticipated delivery dates and other relevant milestones that will be achieved in the upcoming Program Increment. Figure 1 – Example Program Board shows sample output for multiple teams participating in a PI Planning session.

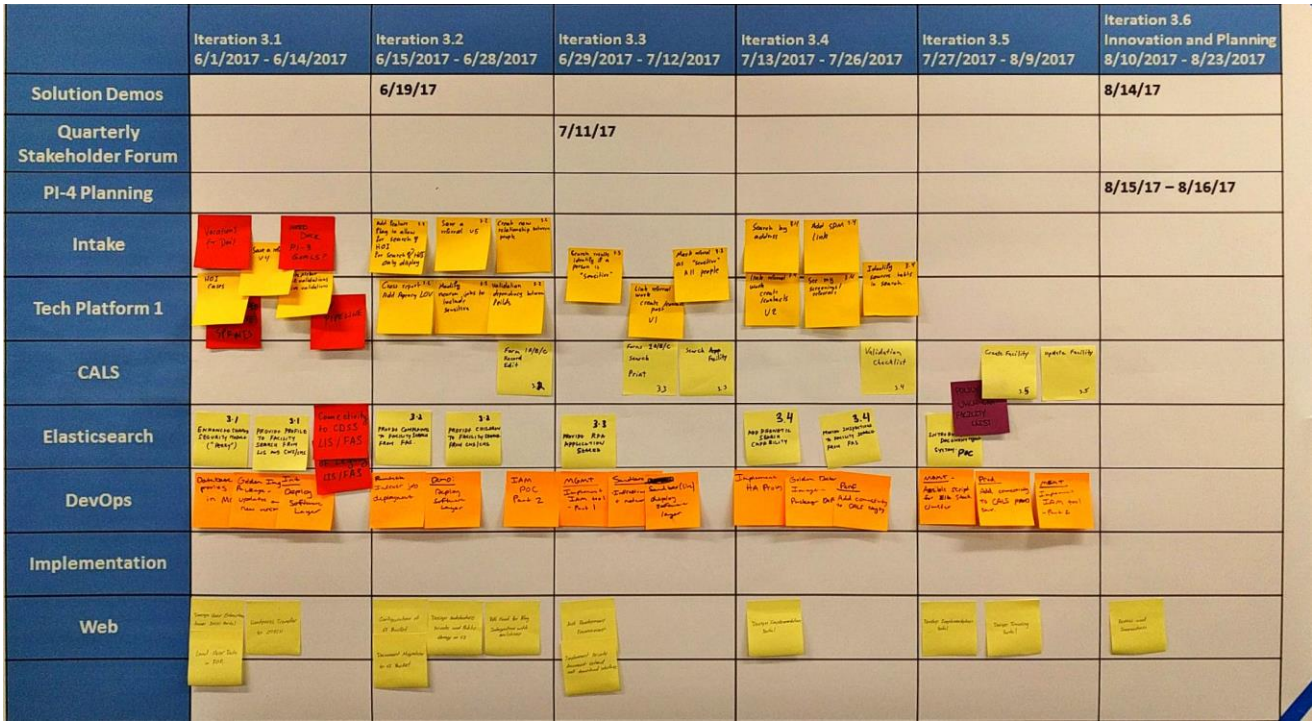

Figure 1 – Example Program Board

At the conclusion of the breakout sessions for the day, the teams will present to the PI Planning participants their PI draft plan, which includes:

- Draft objectives and features by which sprint they intend on delivering the planned features
- Potential risks and dependencies

Project team members are excused and Management (Service Managers, Product Owners, and Scrum Masters) conduct a Review and Problem-Solving session. It is likely that the draft plans present challenges such as scope, resource constraints, and dependencies. During this review and problem-solving meeting, management negotiates scope and resolves these challenges by agreeing to various planning adjustments.

Following the review, the teams will breakout again and perform the following:

- Review the final desired features
- Confirm cross-team dependencies

- Finalize their draft plan, iteration by iteration
- Finalize risks

At the conclusion the additional breakout(s), the teams will conduct a final report out of risks, team objectives and planned features.

Following the report out, the project then conducts a vote of confidence in meeting the program PI objectives. The Release Manager calls for a Fist to Five (3 or higher is equates to acceptance). Any person voting two fingers or fewer is given the opportunity to voice their concern. Assuming that the concerns are resolved, the Program Increment plan, which is represented via the program board, is "locked". While the teams are expected to execute the plan that they just agreed to, it is expected that changes will occur during the PI execution as teams drill down into the identified features or learn of new feature requests. The plan and any changes are reviewed and discussed during the regularly scheduled scrum of scrum meeting, which is described in Section 4.3 Scrum of Scrums.

Prior to starting the PI, each development team should have 5 sprints worth of feature development stories defined, prioritized, and estimated at a high-level in their backlog that aligns with the features listed on the Program Board. Each development team must adhere to the epic naming conventions for PIs as defined here.

Each epic name must be a business name that is recognizable by our stakeholders rather than a technical name and prefixed with the current PI using the format PI-[fill in the number of the program increment with a dash between the PI and the number]. For example: pi-3 screening data. A separate label must be created for each program increment titled "Planned PI- [fill in the number of the PI with a dash between the PI and the number]. For example: planned pi-3. Each feature story identified as part the PI must be associated with an epic and use the "planned pi-x" label. Near-term (iterations 1 & 2) features should be more granularly defined then longer-term (e.g., sprints 4 & 5)

4.3 Innovation and Planning

The last iteration (sprint 6) during the PI is dedicated to Innovation and Planning. Digital Service teams are not expected to plan for development of new features. Instead, they are expected to set aside time for the following types of activities:

- Planning, retrospective, exploration and innovation
- Solution Demonstration
- Inspect and Adapt workshops
- Technical infrastructure and tooling (e.g., new test automation frameworks, new project management tools)

- Continuing education
- Buffer for completing any planned features that didn't get done during the first five iterations
- PI Retrospective
- Work on spikes identified for the next PI

4.4 Scrum of Scrums

The Release Manager meets with the digital service team scrum masters on a recurring basis (twice a week) to review the PI program board (i.e., the plan) and discuss PI milestone, feature and objective progress. Cross-team dependencies, risks, and impediments are also discussed. During this meeting, each service team will discuss the following:

- What did your team accomplish (high-level) since the last meeting?
- What will your team accomplish between now and the next meeting?
- Are there any emerging requirements (i.e., new or further decomposed features)?
    - If so, new post-it notes are added to the program board.
- Will any features not be completed in the planned iteration and need to be moved to future iterations?
    - If so, the post-it note is moved to the appropriate iteration. If de-scoped, they are marked as being de-scoped.
- Any dependencies with other teams?
- Are there any blocking issues/impediments?
- Are there any new risks or updates to existing risks? Do any team risks need to be promoted to project level risks?
- Are we on track for the next Solutions Demo?

4.5 Solution Demo

Solution demonstrations are conducted monthly and provide an opportunity for the digital service teams to showcase their work to date and get feedback from the project and stakeholders. It also helps ensure that integration is occurring across teams on a regular basis. On a quarterly basis, the solution demo will be conducted as part of the CWDS Quarterly Stakeholder Forum, which convenes onsite at the CWDS campus and

provides a general update on project's progress as well as specific topics focused on the individual digital services being developed, program policy, tools, testing, implementation, and other related project information.

4.6 Risk Management

Each of the digital service teams are expected to identify risks during the PI Planning session and post them on the PI program board in the iteration when the risk is most likely to be triggered. All risks posted on the PI program board are initially treated as team risks and will be discussed during team meetings and at the Scrum of Scrum meeting. At any time, if either the Scrum Masters or Release Manager believe that the risk either needs higher visibility or needs assistance in mitigating the risk, they can work with the PMO Risk Manager and escalate the risk to a project-level risk by submitting it to the Project's Risk and Issue forum by submitting a CWDS Candidate Risk Form. It will be then be discussed during the next regularly scheduled Risk and Issue meeting.

4.7 PI Retrospective

The Release Manager will facilitate a PI retrospective meeting with key ART stakeholders (Service Managers, Product Owners, and Scrum Masters) to reflect on what happened during the PI and identify actions for improvement going forward. This meeting is conducted prior to PI Planning. Prior to the meeting, the digital service team leaders are expected to have separate meeting(s) with all of their team members and gather the following:

- What 5 things worked well
- What 5 things didn't work well
- What they will do to improve during the next PI

These items are brought to the PI Retrospective meeting, shared with the group, and organized into common themes. Collectively, the group discusses them and agrees on appropriate action items. Figure 2 – Sample PI Retrospective represents the results from an actual project PI retrospective.

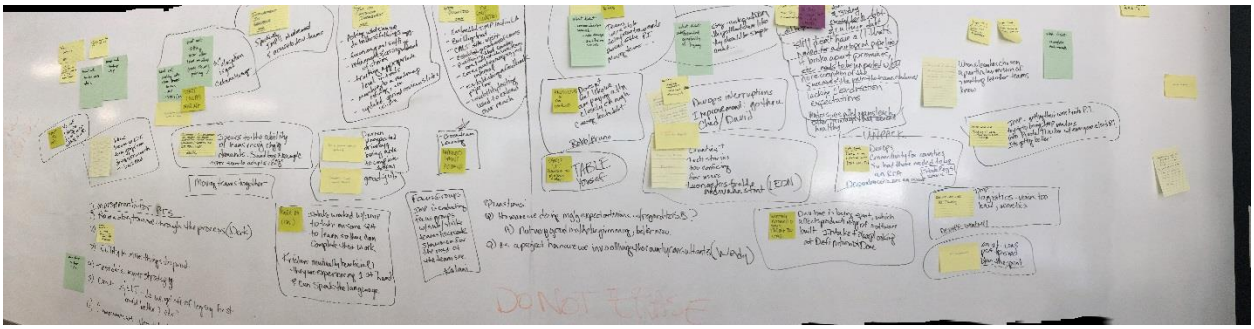

Figure 2 – Sample PI Retrospective

5 Continuous Delivery (CD) Pipeline

The Continuous Delivery (CD) Pipeline (hereinafter referred to as the “pipeline”) describes the project’s approach to providing the continuous releases of software to the end user. The pipeline breaks down the software delivery process into stages that are physically represented by the different environments (e.g., PreInt, Integration, Performance, and Sandbox). Each stage is aimed at verifying the quality of the product from a different perspective and identifying bugs. The pipeline provides feedback to the teams and visibility into all aspects of the continuous delivery process, which includes the building, deploying, testing, and releasing the features. The overall objective of the pipeline is to deliver useful, high-quality, working software to users in an efficient, fast, reliable, and repeatable manner.

Figure 3 – Logical Continuous Delivery Pipeline describes the logical migration path the solution will take.

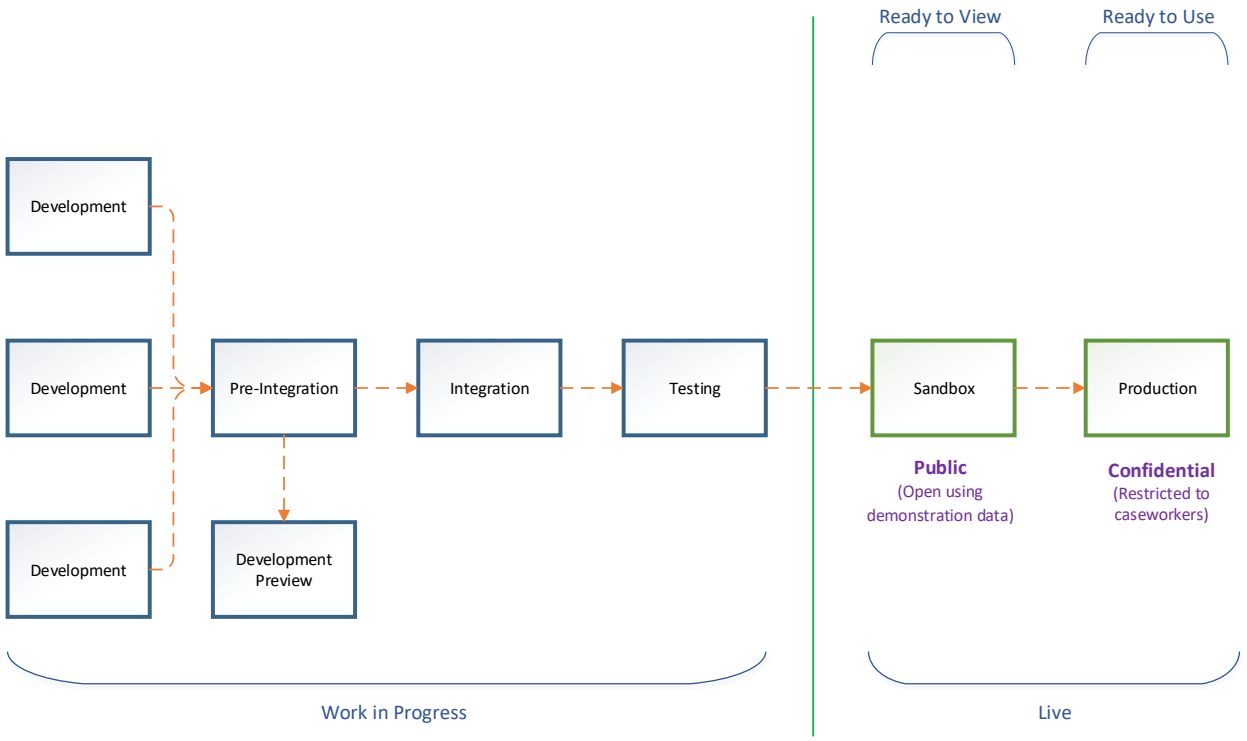

Figure 3 – Logical Continuous Delivery Pipeline

Every change that is made to the application's configuration, source code, environment, or data, triggers the creation of a new instance of the pipeline. The pipeline starts in the Development stage/environment with each development team building, testing and deploying code following a defined continuous integration process. At each subsequent stage in the CD pipeline, the application is tested to ensure that it meets all desired system qualities appropriate for the environment/stage that it is in. Only once the application passes all "Work in Progress" stages can the feature be released to a Live environment (Sandbox or Production).

Figure 4 – Physical Continuous Delivery Pipeline describes the physical migration path that a release candidate will take from one environment to the next on its journey to a Live environment (Sandbox or Production).

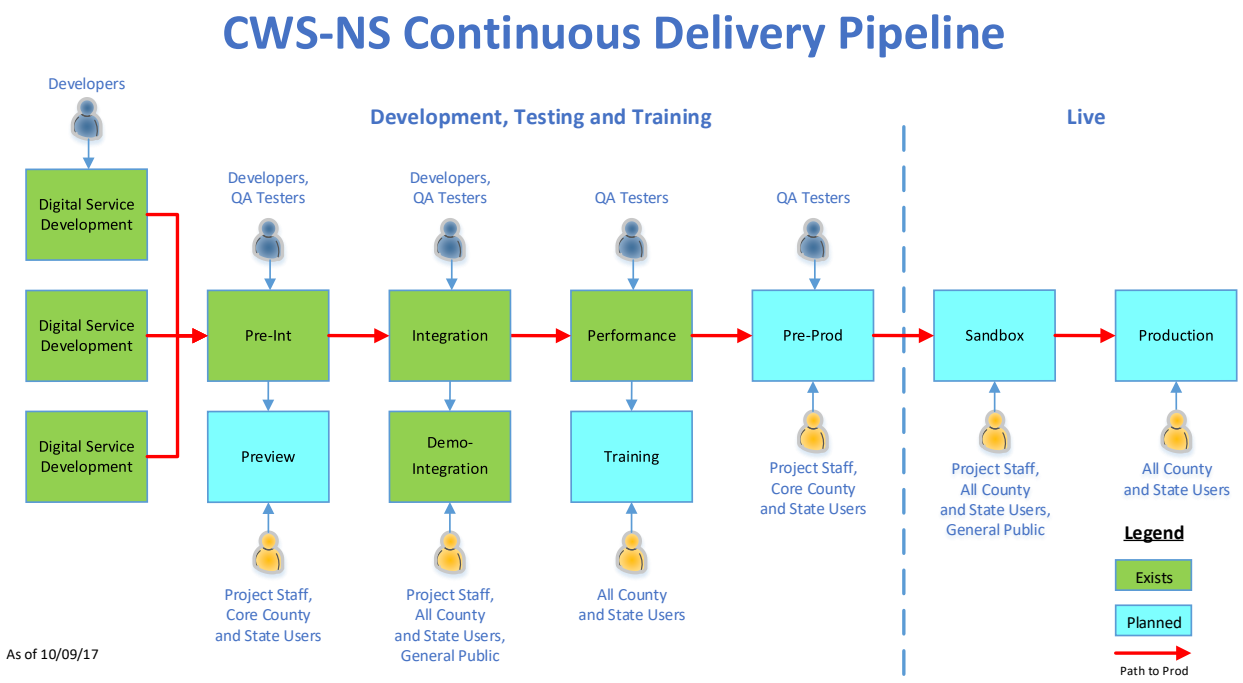

Figure 4 – Physical Continuous Delivery Pipeline

As 10/09/2017, not all environments needed for the pipeline have been configured. Throughout the life of the project additional environments may be created to support multiple candidate releases that are going through pipeline at the same time (e.g., bug release and MVP release). It is also important to know that not all releases must go through Sandbox prior to being released to Production. While code will typically be released to Sandbox prior to releasing it to Production, there are some instances (e.g., bug fixes) when code may be released to both environments simultaneously.

5.1 Continuous Integration (CI)

A key component of the pipeline is the CI process, which is a software development process of integrating developers' work frequently (at least daily) using an automated suite of tools.

There are three important and foundational elements that underpin any Continuous Integration system.

1. **Automated Tests.** The commitment of the development team to produce a comprehensive automated test suite at the unit level and functional level together with their code. It is essential that the automated tests are run with each build to identify potential bugs in a timely manner.
2. **Never break the build.** The goal is that the application must be ready to be built, packaged and deployed on every committed change. This means that broken or untested code should never be committed.

3. **Version Control.** All assets (e.g., source code, automated tests, user stories, configuration files, knowledge transfer material, etc.) must be managed using a using GitHub as the version control system.

There are three key outputs from the CI build process:

1. Automated test results and static code analysis findings will be posted on their Jenkins dashboard page for each component/sub-project
2. If the build is successful (i.e., green), a new Docker image in Dockher Hub is created that can be used to deploy in downstream pipeline environments (e.g., PreInt)
3. The CI results are posted to the teams' Jenkins dashboard page and each of the developers are notified the status of the build (via email or Slack)

5.2 Testing

The types of tests executed in each environment will generally be additive from one to the next, with generally longer running automated tests and manual exploratory tests occurring in downstream environments. For a description of the project's overall testing approach, the different types of testing, and the testing roles, see CWDS Test Management Approach. The specific types of testing that will be conducted during each CD stage is described in Section 5.3 Continuous Delivery Stages for each stage.

5.3 Continuous Delivery Stages

Environment	Development	PreInt	Preview	Integration	Demo-Integration	Performance	PreProd
Purpose	Provide developers an environment to support their Continuous Integration (CI) build process	Provide an opportunity for developers to test their code changes with other digital service teams. It allows the developers to fail fast and identify any potential integration issues	Provide project staff and core county and state users early access to code and the opportunity to provide feedback	Provide digital service teams a full stack environment to conduct automated functional, acceptance, and regression tests. This is also used to conduct Legacy testing	Provide project staff the ability to showcase digital services to core county and state users as well as other stakeholders	Provide project staff and core county and state users early access to code and the opportunity to provide feedback	Provide core county and state users the opportunity to validate release features in an environment that uses production data prior to deploying the candidate release into production
Participants	• Digital service teams	• Digital service teams	• Project staff • Core county and state users	• Digital service teams • QA engineers	• Project staff	• Project staff • Core county and state users	• Project staff • Core county and state users with legacy (e.g., CWS/CMS) production access
Types of testing	• Unit • Functional • Accessibility • Syntax or style checking • Static code analysis • Test coverage analysis • Security testing	• Smoke testing • Acceptance testing • Integration testing	• Usability Testing • Non-scripted feature validation • General feedback	• Smoke testing • Acceptance testing • Integration testing • Regression testing • Legacy testing	• Scripted scenarios	• Smoke testing • Performance Testing	• Smoke testing • Non-scripted feature validation • General feedback
Configuration	• No CWS/CMS access • Limited interface (e.g., Address validation) capability	• No CWS/CMS access • Connects to non-legacy infrastructure (i.e., containers) • Limited interface (e.g., Address validation) capability • Users are added to a LDAP user name and password directory • Uses non-RACF credentials or guest/guest	• User can access using any internet capable device • No CWS/CMS access • Limited interface (e.g., Address validation) capability • Users are added to a LDAP user name and password directory	• User can access using any internet capable device • CWS/CMS access • Limited interface (e.g., Address validation) capability • Uses non-prod RACF credentials	• User can access using any internet capable device • CWS/CMS access • Limited interface (e.g., Address validation) capability • Uses non-prod RACF credentials • Shares the same database with Integration	• User can access using any internet capable device • No CWS/CMS access • Limited interface (e.g., Address validation) capability • Uses prod RACF credentials	• User can access using any internet capable device • CWS/CMS access • Limited interface (e.g., Address validation) capability • Uses prod RACF credentials
Location	Any location	Any location	Any location	Any location	Any location	Any location	Any location

Environment	Development	PreInt	Preview	Integration	Demo-Integration	Performance	PreProd
Data	Mock data that is personally identifiable information (PII) and protected health information (PHI) compliant	Mock data that is personally identifiable information (PII) and protected health information (PHI) compliant	Mock data that is personally identifiable information (PII) and protected health information (PHI) compliant	Mock data that is personally identifiable information (PII) and protected health information (PHI) compliant	Mock data that is personally identifiable information (PII) and protected health information (PHI) compliant	A copy of legacy production database	A copy of legacy production database
Availability	24x7, except for scheduled maintenance	24x7, except for scheduled maintenance	24x7, except for scheduled maintenance	24x7, except for scheduled maintenance	24x7, except for scheduled maintenance	24x7, except for scheduled maintenance	24x7, except for scheduled maintenance
Frequency	Ongoing	Ongoing	Updates are made at the end of every iteration	New features are added to Integration at the end of every iteration	As required	Updates are made at the end of every iteration	Prior to each release to production.
Duration	Ongoing	Ongoing	Ongoing	Ongoing	Varies by demonstration	Ongoing	Varies by release, but will be time-boxed (e.g., 2-3 days)
Release Notes	None	Published in the Preview GitHub repository here	Published in the Preview GitHub repository here	None	None	None	Published in the PreProd GitHub repository
Outcomes	• Completed code • Identification of new bugs • Identification of functionality that could use more automated tests	• Identification of new bugs • Identification of new features • Identification of functionality that could use more automated tests	• Identification of new bugs • Identification of new features • Identification of functionality that could use more automated tests	• Identification of new bugs • Identification of functionality that could use more automated tests	• Identification of new bugs • Identification of new features	• Identification of new bugs • Identification infrastructure enhancements	• Identification of new bugs • Identification of new features • Identification of functionality that could use more automated tests
Feedback	• Bugs are logged in the appropriate digital service team's icebox	• Bugs are logged in the appropriate digital service team's icebox	**Core county and state users:** Feedback is submitted by to email addresses set up by each digital service Feedback is triaged by the development teams • Incidents are logged as bugs in the appropriate digital service team's icebox	• Bugs are logged in the appropriate digital service team's icebox	Feedback is captured during the demonstration session • Incidents are logged as bugs in the appropriate digital service team's icebox • Enhancement requests are captured as new feature stories in the appropriate digital service team's icebox	• Bugs are logged in the appropriate digital service team's icebox	**Core county and state users:** Feedback is submitted by to email addresses set up by each digital service Feedback is triaged by the development teams • Incidents are logged as bugs in the appropriate digital service team's icebox

Environment	Development	PreInt	Preview	Integration	Demo-Integration	Performance	PreProd
			• Enhancement requests are captured as new feature stories in the appropriate digital service team's icebox and looked at as part of each digital service team's story grooming process		and looked at as part of each digital service team's story grooming process		
Training	None	None	Job aids are published in the Digital Services Implementation Portal under the Training /Sandbox folder here **Note**: Publishing of new/updated job aids may lag deployment of new code to the Preview environment	None	Demo material is provided to participants on an as needed basis	None	Draft training material is provided to participants

5.3.1 Development (Build)

Description

This is the development (build) environment for that is unique for each of the digital service teams.

Purpose

The purpose of the Development environment is to support each digital service team's CI build process.

Assumptions

- Automated tests are run as part of the CI build process
- Developers have local containerized instances of Legacy and Postgres databases to work with.

Entry Criteria

- Developers pull down the latest version of all the code from the GitHub repo that they are working on and execute it on their local development environment with their changes.

Deployment Activities

- Once the developer finishes their development and testing on their local development environment, they check their code into their team's GitHub repository and merge their changes into the master branch, which triggers automated CI (build) instructions that are managed using Jenkins. The CI process starts with running through a set of automated unit tests and code analysis. PreIntThe CI results are posted to the teams' Jenkins dashboard page and each of the developers are notified the status of the build (via email or Slack). If the build is successful (i.e., represented as a green circle in Jenkins), a new Docker image is created that can be used to deploy in downstream pipeline environments (e.g., PreInt).

Testing Activities

- The following types of tests are executed as part of each team's automated CI process:
    - Unit
    - Functional
    - Automated security testing
    - Accessibility
- The following types of code analysis checks are executed as part of each team's automated CI process (Back-end teams typically use SonarQube and Front-end teams typically use CodeClimate to execute these tests):
    - Syntax or style checking

- Static code analysis
- Test coverage analysis
- Security testing

Acceptance Criteria

- The build passes (represented as a green circle in Jenkins.

Future Changes

- The Development Teams need to include security testing as part of their CI build process.

5.3.2 Pre-Integration (PreInt)

Description

This is an integrated environment that is using containerized instances of legacy databases with mock data. By design, this environment will be volatile (i.e., updated frequently) as developers push new/updated containers throughout the day.

Purpose

Provide an opportunity for developers to test their code changes with other digital service teams. It allows the developers to fail fast and identify any potential integration issues.

Assumptions

- Automated tests are run manually in PreInt (not part of the deployment process)
- Core County Users will not access PreInt
- PreInt will always contain the mainline version of code (i.e., feature flags are not turned on)

Configuration

- Uses separate containerized instances of Legacy databases (CMS, LIS, FAS) for each digital service team. Does not include the Legacy full stack infrastructure (e.g., CICS and COBOL logic).
- Each digital service team uses its own separate Postgres database instance (e.g., one for CALS and one for Intake).
- All databases are configured with mock (i.e., obfuscated) data.
- Each digital service team creates and uses their own test data for automated tests.
- All Postgres databases are reset each night with a baseline set of seed data. Elasticsearch is updated as part of that process to remove added content.
- Legacy databases are not typically reset each night.
- Authentication uses non-RACF credentials or guest/guest.

Entry Criteria

- Development Acceptance Criteria has been met (i.e., a successful CI build).

Deployment Activities

- Developers manually deploy the latest “green” container(s) to PreInt on an as needed basis

Testing Activities

- Developers manually execute their automated acceptance test scripts.
- Other team resources (e.g., QA, Product Owner) perform manual and exploratory tests.

Acceptance Criteria

- New features pass either automated or manual acceptance tests.
- Code meets or exceeds 70% Test Coverage

Future Changes

- Agree on a process for labeling Docker images as “Ready” for promotion to Int.
- Smoke tests will be automated with each deployment to PreInt to verify that the application and all key services (e.g., Security API, Search API) are working.
- Each team’s CI build process will be configured to use a static code analyzer (e.g., SonarQube or CodeClimate) to achieve a minimum code quality goal. Examples include the following:
    - No Security Vulnerabilities
    - No Blocker or Critical Issues

5.3.3 Integration

Description

This is an integrated environment that is utilizing the full legacy infrastructure and mock data.

Purpose

The purpose of the Integration environment is to validate the work of the entire project team in a full stack environment using automated functional, acceptance, and regression tests as part of each deployment, when possible. Legacy regression testing will likely be manual for the foreseeable future.

Assumptions

- The environment is fully built out and supports the necessary network connections (e.g., VPN access)
- All Legacy system (CMS, LIS, and FAS) testing is done manually
- Each digital service team creates the suite of automated and manual tests that will be run. This includes all scenarios required to validate the features.

- Each digital service team creates their own test data for automated tests
- Integration deployment is done manually on an as-needed basis
- As needed, there are more than one Integration environments to support testing of specific code branches (e.g., Snapshot)
- Legacy database CWSNS1 is shared with multiple Integration environments.
- Core County Users will not be using Integration
- Jenkins deployment scripts are stored in GitHub.

Configuration

- Uses a single instance of each of the Legacy databases (CMS, LIS, FAS), which includes the full stack infrastructure (e.g., CICS and COBOL logic).
- Uses a single Postgres relational database service (RDS) for all front-end digital service teams
- All databases are using mock data
- Postrgres databases are reset with each deployment and reset each night
- Legacy databases (e.g., CWSNS1) are not reset every night
- ElasticSearch is rebuilt with each deployment
- Authentication has the option to either use RACF credentials or guest/guest.

Entry Criteria

- PreInt Acceptance Criteria has been met.

Deployment Activities

- DevOps manually deploys the specified versions of containers to specific environments (e.g., INT, INT02) on an as needed basis. This will be coordinated by the Release Manager.
    - Correct versions/dependencies of containers are deployed together.
- After deploying to Integration a set of smoke tests are run to determine if the deployment was successful and the installed application is operating as expected. This will involve just a few tests exercising functionality that depends on the configuration settings being correct. Ideally, these tests should stop the application and fail the installation or deployment process if the results are not as expected.

Testing Activities

- Once the code has been deployed and passes the smoke tests, the developers will run their team's suite of automated tests. If this cannot be automated to run with the deployment, it will likely be done in the morning after a deployment.
- Legacy testers execute manual legacy tests
- Other team resources (e.g., QA Engineers, Product Owner) perform manual and exploratory tests

Acceptance Criteria

- Passed smoke testing (Yes, No, N/A)
    - At a minimum, must address all applicable architecture components
    - Example (Add a client and search for the client)
- All feature changes must meet the following:
    - Utilizes all required technology platform architecture components – In other words, no stubbed-out code (Yes, No, N/A)
    - Business rules and validation checks implemented (Yes, No, N/A)
    - Passed functional and integration tests – including tests based on user defined use cases (Yes, No, N/A)
    - Passed accessibility testing using an automated accessibility testing tool (pa11y) and passes 18F's Accessibility Checklist Critical Items (Yes, No, N/A)
    - Passed smoke testing (Yes, No, N/A)
    - Passed regression testing (Yes, No, N/A)
    - Passed exploratory testing (Yes, No, N/A)
    - Passed legacy testing (Yes, No, N/A)
    - Test Coverage > 70% (Yes, No, N/A)
    - No open critical bugs (Yes, No, N/A) If yes, identify bug references and provide reason for promoting
    - Meets all "Ready" CWDS pattern library component standards (Yes, No, N/A) If no, explain
    - Infrastructure consistent across all environments (Yes, No, N/A) – this is a DevOps acceptance criteria
    - Passed browser compatibility testing (Yes, No, N/A)
        - Browser Priority 1 (Standards compliant)
            - WebKit
                - Chrome
                - Safari
                - Edge
            - Gecko
                - Firefox
        - Browser Priority 2 (Non-Standards compliant)
            - Microsoft (Legacy)
                - IE v11
    - Product Owner Approved (Yes, No) – N/A is not an option ☺
    - QA Approved (Yes, No, N/A)
    - Demonstrated to Core County Users and feedback incorporated. The demonstration can be done in any environment. (Yes, No, N/A) – If no, explain.

Future Changes

- DevOps will provide non-Devops resources the ability to deploy to Integration.

- DevOps will configure the ability to automatically run the suite of automated tests as part of the deployment to Integration.
- CWS-CMS legacy databases (e.g., CWSNS1) will be configured to reset to the baseline version (CWSNS0).

5.3.4 Preview

Description

This is an integrated environment that is based on PreInt "ready" Docker images. Like PreInt, this environment will be updated frequently. It is configured using containerized instances of databases. The project is treating the core-county users as an extension of our project team and our Preview environment as an extension of our Development/PreInt environment. Therefore, the project is not applying the same scrutiny that it will in any of the Live environments (Sandbox, Prod).

Purpose

The purpose of the Preview environment is provide project staff and core county users an opportunity to preview changes to the application and provide feedback to the development teams.

Assumptions

- The Preview environment is fully built out and supports the necessary network connectivity (e.g., core county access)
- The current promotion timeframe is at the conclusion of each sprint
- There are no Release Notes or Job Aids for Preview

Configuration

- Uses a single instance of each of the Legacy databases (CMS, LIS, FAS), which includes the full stack infrastructure (e.g., CICS and COBOL logic)
- Uses separate Postgres database for each digital service team (e.g., one for CALS and one for Intake)
- All databases are using mock data.
- Postrgres databases are reset with each deployment, but not reset each night
- Legacy databases (e.g., CWSNS1) are not reset
- ElasticSearch is rebuilt with each deployment
- Authentication uses non-RACF unique user names and passwords

Entry Criteria

- PreInt Acceptance Criteria has been met.

Deployment Activities

- DevOps manually deploys the specified containers as coordinated by the Release Manager.

Testing Activities

- Core County Users and project staff use the system and provide feedback to the digital service teams.

Acceptance Criteria

- None – The Preview environment is not on the Continuous Delivery Pipeline towards production.

Future Changes

- Agree to frequency of promoting to Preview sooner than at the end of each sprint (e.g., nightly, as-needed)
- Agree to promotion process (e.g., allow developers the ability to promote, schedule it nightly)

5.3.5 Performance (Perf)

Description

This is an environment that is primarily used to conduct load and performance testing.

Purpose

The purpose of the Perf environment is to conduct and pass tests that meets or exceeds the project defined load and performance benchmarks.

Assumptions

- The environment is fully built out and supports the necessary network connections (e.g., VPN access)
- Contains a copy of CWS-CMS production data
- Until Pre-Prod is built, Performance will be used to support Performance and Pre-Prod testing activities
- There currently are no defined CWS-CARES load and performance benchmarks
- Core County Users will not use Perf
- PreProd and Perf are currently one and the same.

Configuration

- Uses a single instance of each of the Legacy databases (CMS, LIS, FAS), which includes the full stack infrastructure (e.g., CICS and COBOL logic)
- Uses a single integrated Postgres RDS instance for all front-end digital service teams.
- Legacy databases are using a copy of CWS-CMS production data
- Postrgres database is reset with each execution of the performance tests
- Legacy databases (put correct db names here) are reset to CWSNS0 with each execution of the performance tests

- ElasticSearch is rebuilt with each deployment and with each execution of the performance tests
- Only specified users with production access will have access
- Authentication will use SAF with named user id/ password based on production level RACF credentials.

Entry Criteria

- A set of load and performance tests have been created.
- Integration Acceptance Criteria has been met.

Deployment Activities

- DevOps manually deploys the specified containers as coordinated by the Release Manager.

Testing Activities

- Smoke tests are executed
- Development teams execute the load and performance tests
- Security Lead executes the Security Tests
- Tests that require “production” data are executed. This is primarily for tests related search functionality that requires the volume and uniqueness of data that can only be found in production.

Acceptance Criteria

- Passed smoke testing (Yes, No, N/A)
- Passed Information Security Lead’s security tests (Yes, No, N/A)
- Passed functional tests (e.g., search) (Yes, No, N/A)
- Passed load testing (Yes, No, N/A)
- Passed performance testing (Yes, No, N/A)
- Provided summary report of all load and performance test results

Future Changes

- The project will define and develop a baseline set of load and performance benchmarks.
- The project develops a set of obfuscated production test data.

5.3.6 PreProd

Description

This is an environment that exactly resembles the production environment. It is a complete but independent copy of the Production environment, including the databases.

Purpose

The purpose of the PreProd environment is to execute the final set of smoke and security tests before releasing it to one of the Live environments (Sandbox and Production). Another purpose of the Pre-Prod environment is to provide core county and state users the opportunity to validate release features in an environment that uses production data prior to deploying the candidate release into production.

Assumptions

- This environment has not yet been stood up.
- PreProd activities will be conducted in the Performance environment until this environment is available.

Configuration

- Uses a single integrated Postgres RDS instance for all front-end digital service teams.
- Uses a single instance of each of the Legacy databases (CMS, LIS, FAS), which includes the full stack infrastructure (e.g., CICS and COBOL logic)
- Legacy databases are using a copy of production data
- Databases are reset with the execution of the any tests
- ElasticSearch is rebuilt with each deployment and with each execution of the performance tests
- It is certified for secure production access (i.e., only specified users with legacy production access will have access)
- Authentication will use SAF with named user id/ password based on production level RACF credentials.

Entry Criteria

- Performance Acceptance Criteria has been met.

Deployment Activities

DevOps manually deploys the specified containers as coordinated by the Release Manager.

Testing Activities

- Smoke Testing
- Security Testing
- User Feedback

Future Changes

- DevOps will stand this environment up.

5.3.7 Sandbox

Description

This is a Live environment that resembles the Production environment with a few key exceptions:

- No integration with legacy (CWS/CMS or LIS/FAS)
- No system interfaces (e.g., address validation, SDM), which will require significant amount of custom code.

Purpose

The purpose of the Sandbox environment is to allow the public, which includes project staff, county users, internal and external stakeholders, as well as the general public, the ability to access the solution and provide feedback. One of its primary goals is to allow county users the ability to try out the software to help assess when they are ready to adopt the software in their county. In order to support the objectives of the Sandbox, the following design constraints must be applied to the building and configuring of this environment:

- Available to the general public
- Support user data isolation (in other words, users cannot see other users' data)
- Support allowing user to select a role (e.g., supervisor)
- Minimize the amount of unique one-off Sandbox-only code
- Use containerized legacy database instances
- Limited interface capability (i.e., Uses stubbed interfaces)
- Use watermarks or some type of indicator that notifies the user that they are using Sandbox

Assumptions

- Authentication approach has not been determined
- Architecture has not been determined

Configuration

- Uses separate containerized instances of Legacy databases (CMS, LIS, FAS) for each digital service team. Does not include the Legacy full stack infrastructure (e.g., CICS and COBOL logic).
- Uses separate Postgres database instances (e.g., one for CALS and one for Intake).
- All databases are using mock data.
- Each digital service team creates and uses their own test data for automated tests.
- All Postgres databases are reset each night with the baseline set of seed data. Elasticsearch is updated as part of that process to remove added content.
- Legacy databases are not typically reset each night.
- Authentication uses user name/password of guest/guest.
-

Entry Criteria

- Perf/PreProd Acceptance Criteria has been met.

Deployment Activities

- DevOps manually deploys the specified containers as coordinated by the Release Manager.

Testing Activities

- Exploratory Testing. Users will be provided the ability to enter feedback.

Future Changes

- This environment needs to be built.
- Need to agree on how to best limit user's access to other user's data.

5.3.8 Production

Description

This is the final stage/environment in the CD pipeline. This is the place where all code is released to for users directly interact with.

Purpose

The purpose of the Production environment is to support child welfare workers conduct their job.

Assumptions

- The counties will work closely with the Implementation team to determine a timeframe for rolling the solution out to their county.

Key Configuration Components and Settings

- Uses a single integrated Postgres RDS instance for all front-end digital service teams.

Entry Criteria

- Perf/PreProd Acceptance Criteria has been met.

Deployment Activities

- DevOps manually deploys the specified containers per the Release Management Governance process

Testing Activities

- Users provide feedback of any identified bugs, issues, or system enhancement requests

Future Changes

- TBD

5.4 Pipeline Configuration Management (CM)

As builds are promoted through the CD pipeline, it is critical that the project be able to track what versions of code have been deployed to what environment. While the project is looking into enterprise CM tool, DevOps is tracking versions of all containers for each environment here.

6 Releases

In following SAFe®'s ART goal of "Release Any Time" the project would like to achieve frequent delivery of working and fully tested increments of software. This is accomplished via a stream of releases that may happen at any time and are not tied to PIs.

6.1 Release Types

Releases are broken into three different types of releases:

- Major Release – The delivery of a set of features associated with a Digital Service's MVP into production. Each Major Release typically requires Implementation Teams' One-Time Services (e.g., OCM, Training, On-Site Support)

- Minor Release – The delivery of system enhancements into production. Each Minor Release typically does not require Implementation Teams' One-Time Services (e.g., OCM, Training and On-Site Support)

- Bug Release – The delivery of backwards-compatible bug fixes into production. Bug Releases do not require Implementation Teams' One-Time Services (e.g., OCM, Training and On-Site Support)

6.2 Release Version Numbering

Release version numbering will align with the release types using the following version numbering convention MAJOR.MINOR.PATCH.

1) **MAJOR** - Increment the first digit for all releases that include a new initiative (e.g., CANS 1.0) that are ready for statewide rollout. The next planned release will be named CARES 2.0.0, which will include CANS 1.0.
2) **MINOR** - Increment the second digit for all releases that include new features/feature enhancements for already implemented initiatives (e.g., IDM 1.3) or new initiatives that are not ready for statewide rollout (e.g., Hotline 1.0). The next planned release after CARES 2.0.0 will be named CARES 2.1.0, which will include IDM 1.3 and Hotline 1.0

3) **PATCH** - Increment the third digit for patches/hotfixes (i.e., non-planned updates to address system issues that must be deployed prior to a planned release).

The following table describes some possible combinations.

Release Components	Example	Release Numbering Convention
If a Release contains a new initiative ready for statewide roll-out, and new features for a previously deployed initiative, and hotfixes the Release will use the new Major Release logic.	• CANS 1.0.0 • Snapshot 1.4.0 • IDM 1.2.1	• CARES 2.0.0
If a Release does not include a new initiative that is ready for statewide roll-out, but does include new features for a previously deployed initiative and hotfixes, the Release will use the Minor Release logic.	• CANS 1.1.0 • Snapshot fixes for CWS/CMS 8.4	• CARES 2.1.1

6.3 Release Principles

The following are the core principles regarding releases that the project will try to adhere to:

- Will focus on developing user-ready solutions (i.e., MVPs) over trying to meet fixed or arbitrary deadlines
- Will establish a development cadence (12 week Program Increment) that is not tied to releases
- Will develop potentially shippable increments (PSIs) with each iteration
- Will ensure end-user buy-in during design and development of the solution (e.g., deploy code to the Preview environment)
- Will be accountable for delivery of each digital service solution within the project's allocated time and budget

6.4 Release Management (Scope and Schedule)

The scope of a release is largely driven by each digital service teams' Service Manager, who works with their digital service team, which includes a set of core county users, to identify the specific set of features and capabilities that would comprise of a Minimal

Viable Product (MVP). The timeline to develop and test the MVP and its target release date is determined as part of PI planning sessions. The adoption and rollout of the release to county users is managed and coordinated by the Implementation team.

6.5 Release Rollout

As described in the CWS-CARES Statewide Implementation Plan the rollout of a release includes the actual transition of Orgs to the new Digital Services. Implementation activities are triggered when the Digital Services management confirms delivery of an MVP. The duration of an implementation period may vary due to several factors, such as the complexity of the functionality being released and the number of Orgs participating in the Release. While the number of Orgs transitioning within a Release may vary, the Implementation Methodology being utilized remains consistent throughout. Given Los Angeles' large number of users and locations separate consideration will be given to the phasing and timing of its implementation. Additional details on the implementation approach can be found in the CWDS Implementation Plan.

6.6 Release-level Definition of Done (aka Release Readiness Checklist)

For all types of releases (Major and Minor) to any Live environment (Production and Sandbox), the Release Manager will maintain a release readiness checklist, which serves as the Release Definition of Done, that will identify all the criteria required to be completed prior to releasing any software into a Live environment. See Appendix A – Sample Release Readiness Checklist for a sample checklist that describes the release criteria.

6.7 Release Notes

Release Notes will be developed by each digital service team's Product Owner and submitted to the Release Manager to be included on the project Github page in the following format

Release Version # and Date

Digital Service Team

Added

Changed

Fixed

The following is an example for an Intake release to Preview:

Release Version # and Date (v2.4.1 May 10, 2017)

Intake

Added

- **Screener Information**: Added ability to create a new participant.
- **Screener Narrative**: Added ability to capture concerns about children and families.
- **Person Search**: Added ability to search for clients by First Name, Last Name, Date of Birth, and Social Security Number.
- **Person Demographics**: Added ability to capture specific information about people to ensure that the correct people are added to screenings.

Changed

- Configured the system to reset the data (i.e., remove any new data entered by Preview users) every night.
- Modified the disclaimer on the login page.
- Changed the login process to accept unique named users and secured passwords.

Fixed

- Fixed Person Search highlighting.

6.8 Release Governance

The project will adhere to the following release governance process in promoting code to any Live environment (e.g., Production and Sandbox):

For major releases the primary Go/No-Go Decision Making Body will be comprised of the Release Manager, Solution Architect, Implementation Manager, Service Manager(s) that have release-ready digital service content, Executive Leadership Team (ELT), and a county representative (if a county representative is not available, an ELT representative may serve in that role). When CWS-CARES software is ready for a major release, the Release Manager will facilitate a go/no-go meeting. Note: For Minor releases, the Go/No-Go Decision Body will be comprised of the Release Manager, Solution Architect, and Service Manager(s) only and will not require county or ELT participation.

6.9 Rollback Plan

This is a place holder for details regarding the project's rollback approach to specify the processes required to restore the system to its original or earlier state, in the event of failed or aborted implementation.

7 Appendix – Acronyms and Terms

Acronym/Term	Description

Continuous Integration	The practice of building and testing an application on every check-in.
Continuous Delivery	A software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time.
Continuous Deployment	The process in which every change is automatically deployed to production.
Configuration Management	The process by which all artifacts relevant to your project, and the relationships between them, are stored, retrieved, uniquely identified, and modified
Docker	Container platform that is used by enterprises to build agile software delivery pipelines to ship new software features.
Docker Container	Docker containers are used to bundle application components and their dependencies in a tested package that is easy to deploy. They are reserved for use by stateless components, or stateful (data) components that need to be reset often to a baseline state.
GitHub	A web-based Git or version control repository and Internet hosting service that is used by the project.
Jenkins	An open source automation server written in Java that helps to automate the building and deploying software projects as part of the continuous integration build process.
Legacy Testing	The process of validating CWS-CMS legacy functionality. Specifically, it involves ensuring that CWS-CARES changes does not break any existing CWS-CMS functionality.
Load Testing	The process of putting demand on a software system or computing device and measuring its response. Load testing is performed to determine a system's behavior under both normal and anticipated peak load conditions.
Organization (Org)	The logical group of clients that the project's Implementation team is responsible for helping get ready for the delivery of the new Digital Services (DS) or functionality into a live environment. There are 60 Orgs in California that require implementation services: the State as a whole, the 58 counties, and the tribes (Karuk and Yurok) as a whole. Each Org may consist of one or more office locations.
Performance Testing	The process of determining the speed or effectiveness of a computer, network, software program or device. This process can involve quantitative tests done in a lab, such as measuring the response time or the number of MIPS (millions of instructions per second) at which a system functions.
Release Notes	Release notes are documents, which are released as part of the final build that contains new enhancements that went in as part of that release and the known issues of that build. Release notes also feed the process of end-user documentation, user guide and training materials
Security Access Framework (SAF)	CDSS developed and maintained Identity and Access Management (IdAM) system.
Slack	A cloud-based set of team collaboration tools and services.

SonarQube	An open source platform used by development teams to manage source code quality.

8 Appendix A – Sample Release Readiness Checklist

ID	Digital Service	Area	Readiness Item	Environment	Status	Planned Dat	Comments
1	Intake	Features	Snapshot features completed	Integration			
2			Sealed and Sensitive Decision (Yes, No, N/A)	Integration			
3		Smoke Tests	E2E Smoke Tests Passed (Yes, No, N/A)	Integration			
4		Required Architecture	Utilizes all required technology platform architecture components (Yes, No, N/A)	Integration			
5		Business Rules	Business rules and validation checks implemented (Yes, No, N/A)	Integration			
6		Functional and Integration	Passed functional and integration tests (Yes, No, N/A)	Integration			
7		Accessibility	Passed accessibility testing using an automated accessibility testing tool (Yes, No, N/A)	Integration			
8			Passed 18F's Accessibility Checklist for Critical Items (Yes, No, N/A)	Integration			
9		Regression Testing	Passed regression testing (Yes, No, N/A)	Integration			
10		Exploratory Testing	Passed exploratory testing (Yes, No, N/A)	Integration & Performance			
11		Legacy Testing	Passed legacy testing (Yes, No, N/A)	Integration			
12		Test Coverage	Test Coverage > 70% (Yes, No, N/A)	Integration			
13		OWASP Zed Attack Proxy (ZAP)	Passed ZAP testing (Yes, No, N/A)	Integration	N/A		
14		Code Climate	• No Bugs • No Security Vulnerabilities • No Blocker or Critical Issues	Integration			
15		Performance Testing	Performance Testing Completed and Passed	Performance			
16		Critical Bugs	No open critical bugs (Yes, No, N/A)	Integration			
17		Bug Workarounds	If any bugs, are the identified workarounds	Integration			
18		Pattern Library	Meets all pattern library standards (Yes, No, N/A)	Integration			
19		Browser Compatibility	Passed browser compatibility testing (Yes, No, N/A)	Integration			
20		Product Owner	Product Owner Approved (Yes, No)	Integration			
21		QA Engineer	QA Approved (Yes, No, N/A)	Integration			
22		Core County User Feedback	Demonstrated to Core County Users and feedback incorporated. (Yes, No, N/A)	Integration			
23		Release Notes	Release Notes completed	Performance			
24	Technology Platform 1	Perf/Pre-Production	Replication and ElasticSearch configured	Performance	Yes		
25		Testing	Performance Testing Completed and Passed	Performance			
26		SonarQube	• No Bugs • No Security Vulnerabilities • No Blocker or Critical Issues	Integration			
27		Test Coverage	Test Coverage > 70% (Yes, No, N/A)	Integration			
28		Acceptance Testing	No Must-Fix Bugs	Integration			
29		Production Environment	Replication and ElasticSearch configured	Production			
30	DevOps	Perf/Pre-Prod Environment	Perf/Pre-Prod Environment Configured	Performance			
31			Perf/Pre-Prod Environment Certified	Performance			
32		RunDeck Monitoring	RunDeck Monitoring has been deployed to all higher-order environments	Integration and Performance			
33		Infrastructure	All higher-order environments (PreInt, Integration, and Perf) have consistent infrastructure tool versions.	Performance			
34		Production Environment	Production Environment Configured	Production			
35			Production Environment Certified	Production			
36	Security	Security Testing	Conduct Security tests	Performance			