Republic Services, Inc.

Amazon Web Services Implementation – How to Guide

*InfoPro Web Applications*

Version *1.0*

*June 20th, 2025*

Prepared by:

*InfoPro Development team*

*18500 N. Allied Way*

*Scottsdale, AZ 85050*

[Revision History 3](#_Toc202254462)

[Document Naming 3](#_Toc202254463)

[1. GitHub 4](#_Toc202254464)

[1.1. Branching Strategy 4](#_Toc202254465)

[1.2. GitHub Actions 5](#_Toc202254466)

[1.2.1. Deploying to Production 5](#_Toc202254467)

[1.2.2. Rollback Deployment to Production 6](#_Toc202254468)

[1.2.3. Deploying to Lower Environments 7](#_Toc202254469)

[2. Amazon Web Services 8](#_Toc202254470)

[2.1. Setup of Accounts 8](#_Toc202254471)

[2.2. Password Management 8](#_Toc202254472)

[2.3. Logging 8](#_Toc202254473)

[2.4. Running Local 10](#_Toc202254474)

[Appendices 13](#_Toc202254475)

[Appendix A – Document Content Questions 13](#_Toc202254476)

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Change Description** | **Change Owner** | **Date** |
| 1.0 | Initial Version | Jeff Sprong |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Document Naming

All documentation shall follow the below document naming convention.

Document Name starts with INF for InfoPro followed by an underscore (\_) followed by a 4-char code HWTO/PROJ/VEND (How To/Projects/Vendor) followed by an underscore (\_) and finally a module that the document is going to address.

Document Name: INF\_HWTO\_AWS.docx

# GitHub

# Branching Strategy

A diagram of a branch

AI-generated content may be incorrect.

We shall create/maintain several different branches as part of our software development life cycle.

For each enhancement request we shall create one or more user stories which translates to branches within GitHub i.e., each user story would be a branch within GitHub.

Currently there are two teams creating branches. Team name branches have been developed to be used by each team for merging current user stories. When a deployment is planned, both teams will merge their work if both teams are ready. If only one team is ready to deploy like a production issue, the team will merge their work and deploy. Both teams will not be dependent on one another for deployments.

The story and team branches can be deployed to both the DEV and QA environments for testing.

For each release we shall create something called Release branch from the Master branch. Once the release is completed that branch shall be deleted as the code shall be captured in the Master branch and tagged as well.

Once the development is complete and when the team is ready to perform unit testing, we shall merge the team branches into the Release branch. Testing of the Release branch can be done in both DEV and QA environments.

Once all the unit testing is completed the Release branch shall be merged into the Develop branch which can also be deployed to the DEV and QA environments for testing.

When we are ready to go to production, we shall deploy the Develop branch to the production environment. Once the production deployment is complete the source code from the Develop branch shall be merged into the Master branch.

The same process is repeated for all the enhancement requests.

# GitHub Actions

GitHub actions will be utilized when deploying branches to both lower environments and production.

# Deploying to Production

Only specific employees that are not associated with writing or testing the code will have access to deploy to production. AD groups have been created and implemented in GitHub to allow access. This access shall be granted in GitHub from individuals that have been approved to deploy.

To deploy in production, the user must use GitHub actions. After clicking ‘Run Workflow’ a pop up will appear where the user will enter the branch name and the environment for deployment. When the user selects ‘prod’ they must also enter the Service Now change ticket associated with the deployment. Once the user starts the workflow, Service Now is checked to ensure that the change ticket is in ‘Work in Progress’ status. This status verifies that all approvals have been obtained for the ticket and that the work has started.

A screenshot of a computer

AI-generated content may be incorrect.

# Rollback Deployment to Production

If a rollback is needed, GitHub actions will be used. In actions, the user will select the ‘rollback to previous’ workflow:

A screenshot of a chat

AI-generated content may be incorrect.

The user shall select was branch and environment to perform the rollback and click the ‘Run workflow’ button:  
  
  
A screenshot of a chat

AI-generated content may be incorrect.

# Deploying to Lower Environments

Only specific developers and QA personnel that are assigned to teams that work on the InfoPro applications will have access in GitHub to deploy to the lower environments.

Lower environments consist of both DEV and TEST. Each environment will have multiple instances for teams to use. Naming of these environments shall be in numerical order (dev1, dev2, dev3, etc.).

To deploy in lower environments, the user must use GitHub actions. After clicking ‘Run Workflow’ a pop up will appear where the user will enter the branch name and the environment for deployment. Note: the user will not have to enter a change ticket number as required for deployment to production.

A screenshot of a chat

AI-generated content may be incorrect.

If a user attempts to deploy code using GitHub Actions and is not a part of a specific AD Group allowed to deploy, the deployment will be terminated, and an error will appear in the deployment log:

A screenshot of a computer

AI-generated content may be incorrect.

# Amazon Web Services

# Setup of Accounts

Two accounts were created in AWS. Both **infopro-nonprod** and **infopro-prod** will be used with the InfoPro applications:

**A screenshot of a computer

AI-generated content may be incorrect.**

# Password Management

CBS and InfoPro as400 connection username and passwords are stored in AWS Secrets Manager. All other configurations are stored in Parameter Store.

All secrets are stored under the secret name/secret/{environment}/csweb2/ where the environment value is one of devN|testN|prod

All parameter store values are prefixed with /config/{environment}/csweb2/

For example, config/dev/csweb2/health/check is the parameter name for the health check value success

To have Spring inject these values into a Java bean using annotations, use the @Value() annotation.

So, to inject the health.check value into a bean we would use @Value("${health.check}")

# Logging

To check logs in AWS, the user must select which account to view the logs in:

A screenshot of a computer

AI-generated content may be incorrect.

User needs to navigate to CloudWatch and select Log Groups. User can the select which group to view logs:

A screenshot of a computer

AI-generated content may be incorrect.

Once group selected user can select which log stream to view:

A screenshot of a computer

AI-generated content may be incorrect.

User can then view the log:

A screenshot of a computer

AI-generated content may be incorrect.

# Running Local

Developers will have the ability to run the applications locally during the development phase. Developers will use Amazon Web Services Access Keys when staring the applications locally and will not use any passwords when initiating the run commands.

To run the application locally you need to start the backend application then the front end. First, you need to refresh your AWS credentials.

*Prerequisite* - you must have an entry in your hosts file like so:

127.0.0.1 local.csweb.repsrv.com

Refresh your AWS credentials locally. Login to the AWS console and copy the Access Keys:

*Windows*

A screenshot of a computer

AI-generated content may be incorrect.

*Mac OS*

A screenshot of a computer

AI-generated content may be incorrect.

Copy the three SET/EXPORT commands and paste in a local terminal shell (the same one you will be running the backend in)

Start the backend

Run these commands in the same terminal as step 1:

\*\*\*\* NOTE: the example below reads property configs from AWS env 'dev1'

MacOS

export CSWEB\_ENV=dev1 && export CSWEB\_REGION=us-east-2

mvn clean install tomcat7:run

Windows

set "CSWEB\_ENV=dev1" && set "CSWEB\_REGION=us-east-2"

mvn clean install tomcat7:run

Start the front end

Run this command in a new terminal making sure PWD you are in is the front end module:

ng serve --open --configuration=local --host local.csweb.repsrv.com --base-href /

Appendices

Appendix A – Document Content Questions

Found any issues within the document, kindly reach out to one of the DLs to have the content updated.