



Experience the commitment®

CGI CA—> CAREMAIL PROJECT CONFIGURATION MANAGEMENT PLAN

TABLE OF CONTENTS

1 THE CI/CD PROCESS.....	2
1.1 CODE AND COMMIT	2
1.2 CHECKOUT	3
1.3 BUILD	4
1.4 TEST	5
1.5 PACKAGE	6
1.6 DEPLOY	7
1.7 INSTALLATION ON A NEW HOST	8

I THE CI/CD PROCESS

This document discusses Continuous Integration (CI) and Continuous Deployment (CD), a process that includes everything related to the automation of the pipeline from which a developer adds changes to a central repository until the code is moved to production.

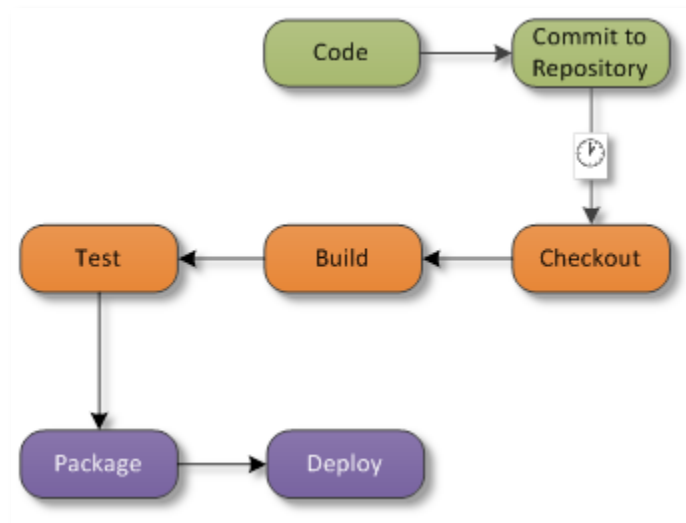


EXHIBIT 1: THE APPLIED CI/CD PROCESSES

Exhibit 1 shows the CI/CD process used to develop the POC. It starts when developers write code and submit it to the centralized repository. Code changes require the code to be checked out before new code is built, tested, packaged, and deployed through an automated process. The following subsections provide details on these CI/CD steps.

1.1 CODE AND COMMIT

Developers used several tools to develop the components of the POC such as Web Apps, Microservices, and Databases. These tools allow developers to easily write the code, debug it, and test it, then directly commit it to the repository, and include:

1. Design the UI
2. Reuse NPM packages
3. Create Web Apps (AngularJS, NGINX)

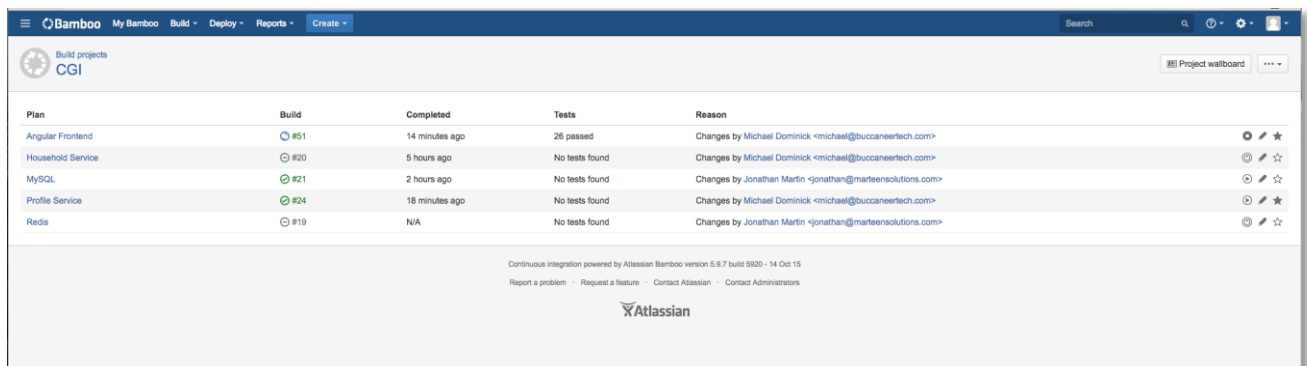
4. Create Microservices (Java, Spring Boot, Java Mail API)
5. Create database entities (MySQL)
6. Link the components via messages (Jersey RESTful Web Services)
7. Perform unit testing (Grunt, Jasmine, Junit, Karma)
8. Commit the web components to a repository (GitHub)

1.2 CHECKOUT

CGI selected Bamboo for automated building, testing, and deployment. Bamboo facilitates deployment projects and environments. A deployment project holds the software project to be deployed, including releases that have been built and tested, and the environments to which releases are deployed.

We defined five plans in Bamboo that it uses to check out the newly pushed elements in the repository every three minutes:

1. Angular Frontend plan
2. Household Service plan
3. MySQL plan
4. Profile Service plan
5. Redis plan



Plan	Build	Completed	Tests	Reason
Angular Frontend	#51	14 minutes ago	26 passed	Changes by Michael Dominick <michael@buccaneertech.com>
Household Service	#20	5 hours ago	No tests found	Changes by Michael Dominick <michael@buccaneertech.com>
MySQL	#21	2 hours ago	No tests found	Changes by Jonathan Martin <jonathan@martensolutions.com>
Profile Service	#24	18 minutes ago	No tests found	Changes by Michael Dominick <michael@buccaneertech.com>
Redis	#19	N/A	No tests found	Changes by Jonathan Martin <jonathan@martensolutions.com>

Continuous integration powered by Atlassian Bamboo version 5.9.7 build 5920 - 14 Oct 15
[Report a problem](#) - [Request a feature](#) - [Contact Atlassian](#) - [Contact Administrators](#)

EXHIBIT 2: BAMBOO BUILDING PLANS

1.3 BUILD

Plan contents vary. For the Angular Frontend plan for example, the plan includes the following three stages:

1. Build and test frontend
2. Build and push frontend container
3. Deploy

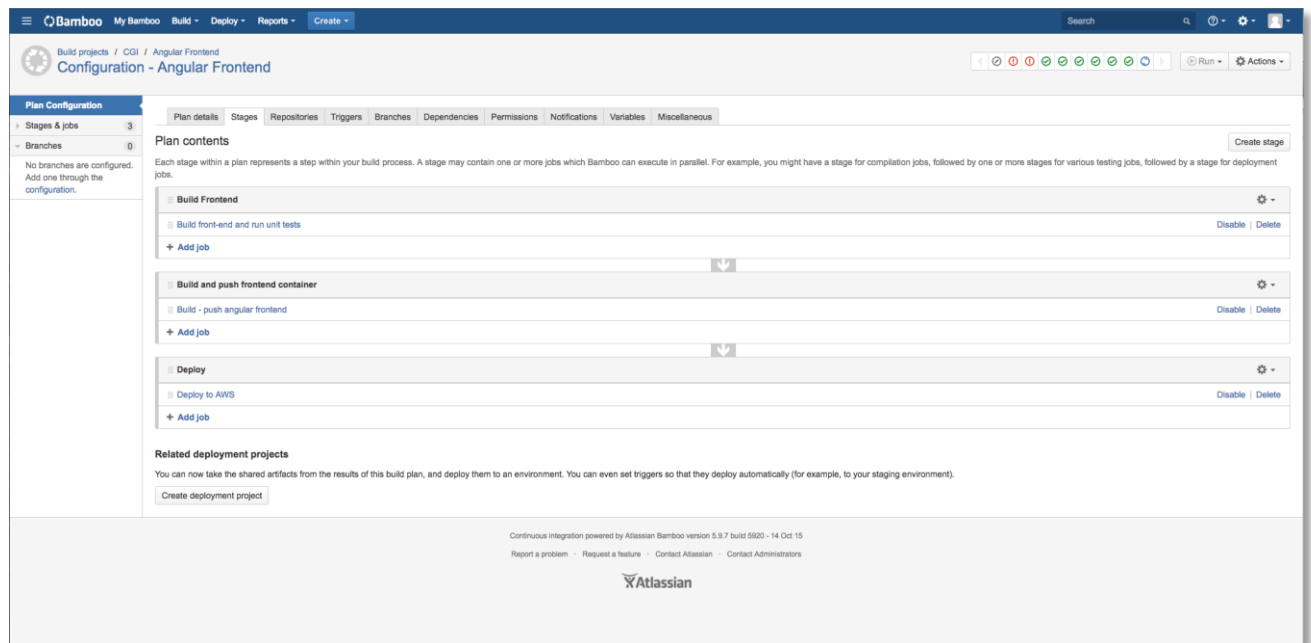


EXHIBIT 3: STAGES WITHIN A PLAN

Each plan is composed of multiple tasks. For example, the Build-Push Angular Frontend plan contains the following tasks:

1. Checkout default repository
2. Docker compose build Angular
3. Push Angular image up to quay

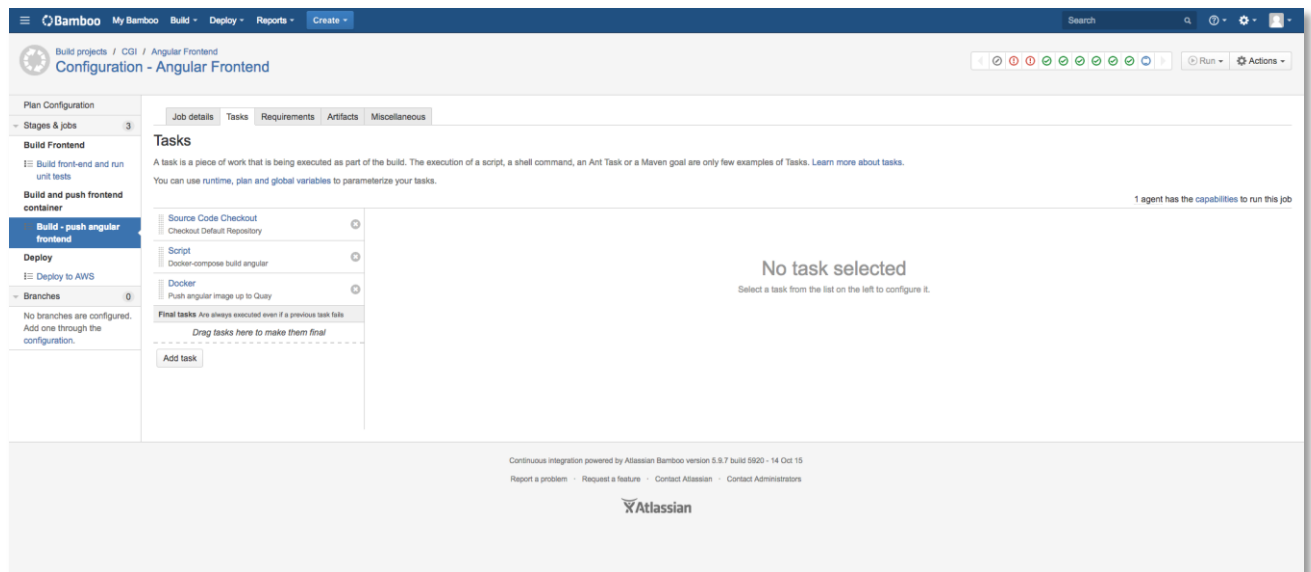
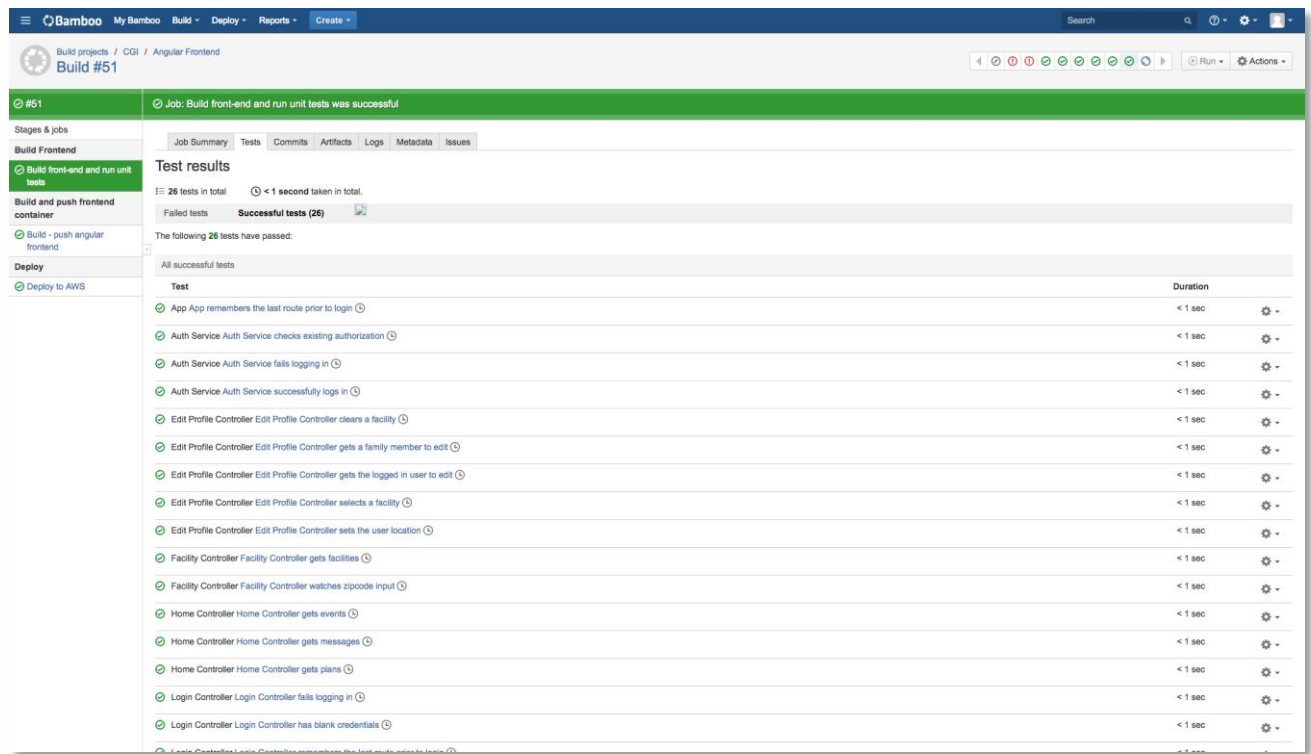


EXHIBIT 4: TASKS FOR BUILD — PUSH ANGULAR FRONTEND

1.4 TEST

Test runs were automated using Bamboo by defining the steps in each test plan. For example, the Build Frontend test included tests such as the following:

1. App remembers last route prior login
2. Auth Service checks existing authorizations
3. Auth Service fails login
4. Etc.



Test	Duration
App App remembers the last route prior to login	< 1 sec
Auth Service Auth Service checks existing authorization	< 1 sec
Auth Service Auth Service fails logging in	< 1 sec
Auth Service Auth Service successfully logs in	< 1 sec
Edit Profile Controller Edit Profile Controller clears a facility	< 1 sec
Edit Profile Controller Edit Profile Controller gets a family member to edit	< 1 sec
Edit Profile Controller Edit Profile Controller gets the logged in user to edit	< 1 sec
Edit Profile Controller Edit Profile Controller selects a facility	< 1 sec
Edit Profile Controller Edit Profile Controller sets the user location	< 1 sec
Facility Controller Facility Controller gets facilities	< 1 sec
Facility Controller Facility Controller watches zipcode input	< 1 sec
Home Controller Home Controller gets events	< 1 sec
Home Controller Home Controller gets messages	< 1 sec
Home Controller Home Controller gets plans	< 1 sec
Login Controller Login Controller fails logging in	< 1 sec
Login Controller Login Controller has blank credentials	< 1 sec

EXHIBIT D: TEST STEPS AND RESULTS

1.5 PACKAGE

We have chosen Docker containers as our packaging tool because it allows us to package the application with all of its dependencies into a standard unit that will run the same way, regardless of the environment it is running in. Docker containers wrap up pieces of software in a complete file system that contains everything it needs to run including code, runtime, system tools, system libraries, and anything that can be installed on a server.

This POC was tested using three different provisioning configurations:

1. Provisioning locally with Docker Machine (Virtualbox)
2. Provisioning locally with Docker Machine (VMware Fusion)
3. Provisioning remotely with Docker Machine (AWS)

Docker images were then built and prepared for deployment, as follows:

1. The repository was first cloned.

2. From within the Docker folder, the Docker-compose build was run.
3. Resulting images were checked.

1.6 DEPLOY

We also decided to use Docker Machine and Docker Compose to handle the provisioning and deployment aspects of our application. Using Docker Machine, we are able to provision a single Docker host or a cluster (Docker swarm) on any of the IaaS providers shown below, as well as many private cloud infrastructures. This gives us the freedom to deploy virtually anywhere without being tied to a single provider.

We selected Amazon Web Services (AWS) as the IaaS provider for the POC due to team experience and top ranking among IaaS providers.

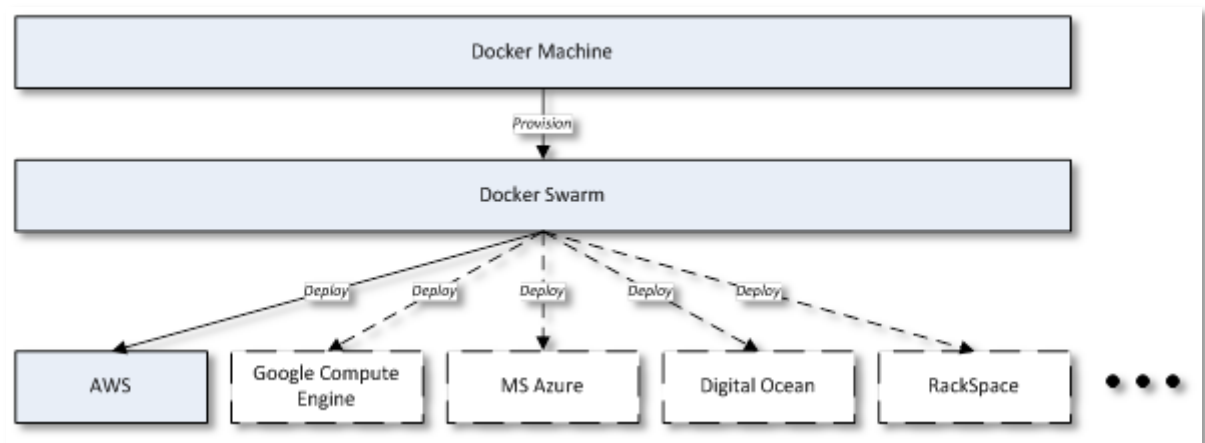


EXHIBIT 6: DEPLOYMENT WITH DOCKER

The Docker deployment script on AWS was configured in Bamboo to automate the deployment process, as shown in Exhibit 7.

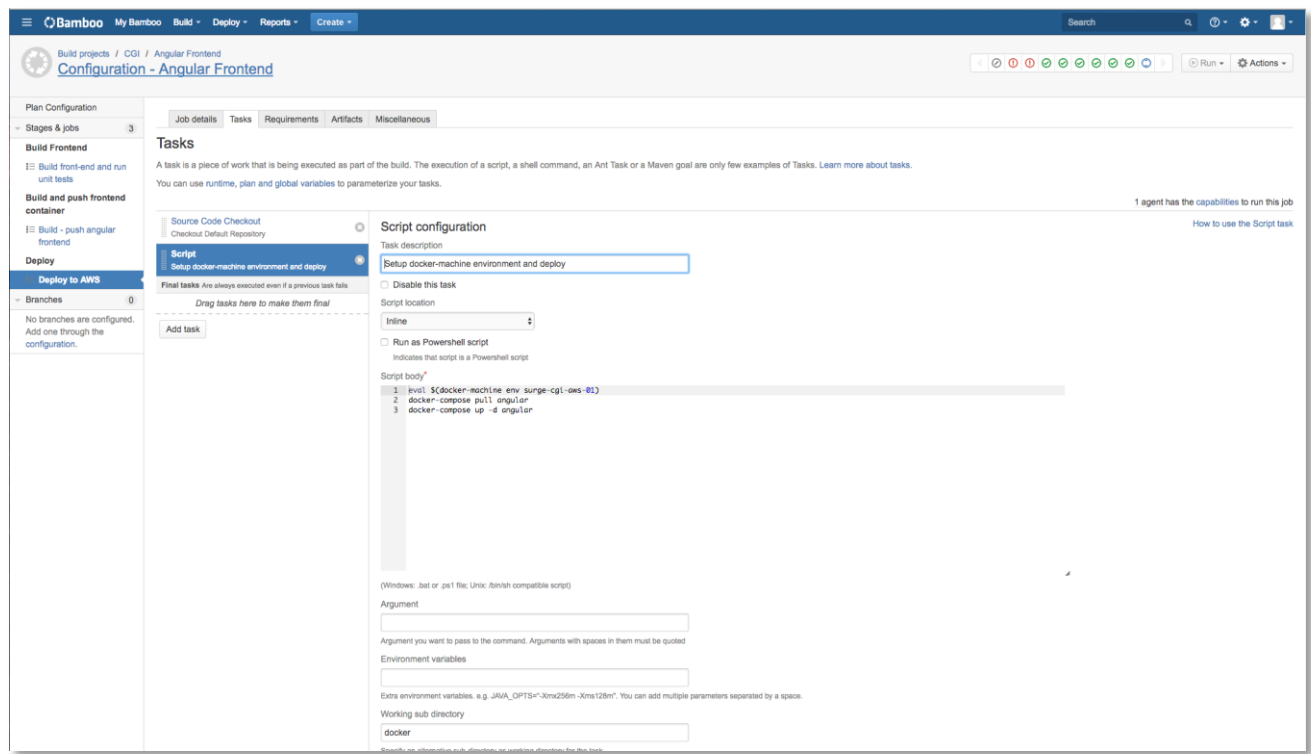


EXHIBIT 7: DEPLOYMENT SCRIPT

1.7 INSTALLATION ON A NEW HOST

Distributed applications consist of many small applications that work together. Docker transforms these applications into individual containers that are linked together. Instead of having to build, run, and manage each individual container, Docker Compose allows us to define a multi-container application with all of its dependencies as a single file and spin it up in a single command. The application's structure and configuration are held in a single place, which makes spinning up applications simple and repeatable.

Installing the application on any new machine only requires Docker and Docker-Compose to be installed to build and deploy the application. All other dependencies are built into the Docker containers. The new host simply needs to be identified in Docker along with the packages to be deployed, along with the defined configuration.