Configuration Management

*Team CAtS (Yihong Zou, Huairui Qi, Siqi Lu, Yun-Hsuan Chang)*

[**Introduction**](#_c05y7x58hoc7) **1**

[**System Overview**](#_fb4xl3ut05i2) **2**

[**Approach**](#_38ta77eloppk) **2**

[Method and Tools](#_h7eyaz1lwlep) 2

[Roles](#_xf6v7plphkhc) 3

[**Configuration Management Administration**](#_62j8mr2rvlkc) **3**

[Configuration Identification](#_wogps1n3fvna) 3

[Configuration items](#_mobxkjschn) 3

[Configuration Control for Different Types of CIs](#_vt1v9uf3k8bu) 4

[Version Controls](#_cnlcgxs9kcp0) 4

[Configuration Control for Cloud Infrastructure](#_onps9bbwmfye) 5

[Configuration Management Process](#_fj1fsopog6sk) 6

# Introduction

The Configuration Management Plan (CMP) aims to provide the technical and administrative guidance for managing the configuration items (CI) in the Continuous Authorization Service (CAS) project which is to be placed under configuration control. This document defines the project’s structure and tools for identifying, defining, and baselining CIs, as well as controlling modifications and releases of CIs.

# System Overview

Application security is a living process that must constantly be addressed throughout the application lifecycle. This requires continuous security assessments at every phase of the Software Development Lifecycle (SDLC). The idea is that, whenever new code is merged into the current repository, the security test will be conducted, either locally or globally, to ensure the security of the software application.

Since a completely automated security evaluation service is not offered on commercial or open source platforms, we are going to create a web-base Continuous Authorization Service (CAS) that automates the whole security testing process. The CAS system can let the user customize their security testing preference, and help extract the user-focused security issue from multiple automated static testing tools. Users do not need to look through every testing report or configure every testing tool by themselves. To enable the members within a team to track the others' progress, users could share the projects with known users and push issues to Github.

In addition to the CAS web service, a plugin which enables the integration of CAS and continuous integration tools will also be a part of the final deliverables. During the development stage, the primary CI tool to integrate will be Jenkins. In the future, CAS wishes to support more continuous integration tools, such as TeamCity, Travis CI, Bamboo, etc.

# Approach

## **Method and Tools**

|  |  |
| --- | --- |
| **Methods** | **Tools** |
| Code Management | Github |
| Document Management | Wiki |
| Progress Management | Git |

## **Roles**

|  |  |
| --- | --- |
| **Name** | **Roles** |
| Yihong Zou | Team leader & Software developer |
| Huairui Qi | Architecture manager & Software developer |
| Siqi Lu | Scrum manager & Software developer |
| Yun-Hsuan Chang | Quality manager & Software developer |

# Configuration Management Administration

## **Configuration Identification**

### **Configuration items**

Configuration identification process consists of selecting proper CIs of the system, and

setting and maintaining baselines of each CI throughout the CAS development lifecycle.

CIs shall include:

● All CAS related requirement documents

● All CAS related design documents

● All CAS related testing plans and results

● All development code required to build the CAS components

● All CAS related data and configuration files

● All CAS related cloud infrastructures

Configuration baselines shall be established progressively according to the current

development phase. Detailed information of each baseline shall be tracked and

maintained using the methods and tools described in Section 3.

## **Configuration Control for Different Types of CIs**

### **Version Controls**

We show the version control on code components in this section. The tool we use for the version control of code components is Git. The version control process for the CAS

project is defined as follows:

* One central repository for the CAS project is established to hold the current and historical versions of all configuration code objects.
* One baseline branch which is the master branch.
* Separate branches created for different features of the CAS project from the master branch.
* After the local development and testing work is completed, the developer should first merge from the remote master branch and resolve conflicts.
* After conflicts are resolved, the developer can start a pull request to the master branch.
* A pull request must be reviewed and approved by other developers and clients before merging into the master branch.

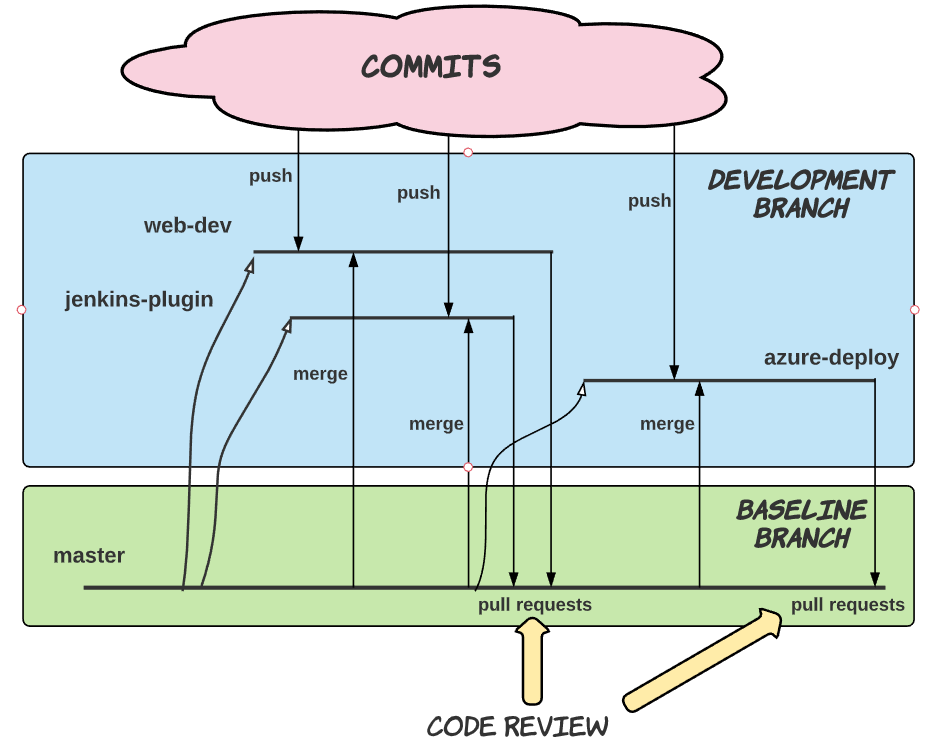


Figure 1: Version Control

### **Configuration Control for Cloud Infrastructure**

CAS project mainly uses Azure and as its cloud infrastructure. The cost of the cloud resource usage should be properly managed with a standard process.

* Use EC2 on demand instances for AWS and Azure Virtual Machine for Azure to initialize the cloud infrastructure.
* Estimate the usage of cloud infrastructure resources for the next working week.
* If there is no need for the cloud resource, the instance should be stopped by the developers for budget management purposes.

### **Configuration Management Process**

The configuration management process includes version control process and Continuous Integration/ Continuous Deployment process. Figure 2 shows an overview of our configuration management process.

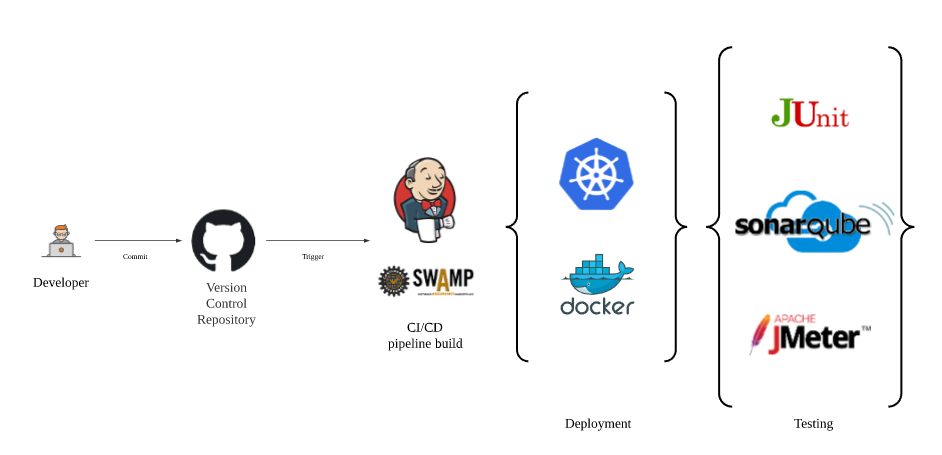


Figure 2: Configuration Process