# Project Configuration

The following sections describe the main part of Zuul’s configuration. All of what follows is found within files inside of the repositories that Zuul manages.

## **Security Contexts**

When a system administrator configures Zuul to operate on a project, they specify one of two security contexts for that project. A config-project is one which is primarily tasked with holding configuration information and job content for Zuul. Jobs which are defined in a config-project are run with elevated privileges, and all Zuul configuration items are available for use. Base jobs (that is, jobs without a parent) may only be defined in config-projects. It is expected that changes to config-projects will undergo careful scrutiny before being merged.

An untrusted-project is a project whose primary focus is not to operate Zuul, but rather it is one of the projects being tested or deployed. The Zuul configuration language available to these projects is somewhat restricted (as detailed in individual sections below), and jobs defined in these projects run in a restricted execution environment since they may be operating on changes which have not yet undergone review.

## **Configuration Loading**

When Zuul starts, it examines all of the git repositories which are specified by the system administrator in Tenant Configuration and searches for files in the root of each repository. Zuul looks first for a file named zuul.yaml or a directory named zuul.d, and if they are not found, .zuul.yaml or .zuul.d (with a leading dot). In the case of an untrusted-project, the configuration from every branch is included, however, in the case of a config-project, only a single branch is examined. The config project branch can be configured with the tenant configuration tenant.config-projects.<project>.load-branch attribute.

When a change is proposed to one of these files in an untrusted-project, the configuration proposed in the change is merged into the running configuration so that any changes to Zuul’s configuration are self-testing as part of that change. If there is a configuration error, no jobs will be run and the error will be reported by any applicable pipelines. In the case of a change to a config-project, the new configuration is parsed and examined for errors, but the new configuration is not used in testing the change. This is because configuration in config-projects is able to access elevated privileges and should always be reviewed before being merged.

As soon as a change containing a Zuul configuration change merges to any Zuul-managed repository, the new configuration takes effect immediately.

## **Regular Expressions**

Many options accept literal strings or regular expressions. In these cases, the regular expression matching starts at the beginning of the string as if there were an implicit ^ at the start of the regular expression. To match at an arbitrary position, prepend .\* to the regular expression.

Zuul uses the RE2 library which has a restricted regular expression syntax compared to PCRE.

Some options may be specified for regular expressions. To do so, use a dictionary to specify the regular expression in the YAML configuration.

For example, the following are all valid values for the job.branches attribute, and will all match the branch “devel”:

**- job:**

**branches: devel**

**- job:**

**branches:**

**- devel**

**- job:**

**branches:**

**regex: devel**

**negate: false**

**- job:**

**branches:**

**- regex: devel**

**negate: false**

## **Encryption**

Zuul supports storing encrypted data directly in the git repositories of projects it operates on. If you have a job which requires private information in order to run (e.g., credentials to interact with a third-party service) those credentials can be stored along with the job definition.

Each project in Zuul has its own automatically generated RSA keypair which can be used by anyone to encrypt a secret and only Zuul is able to decrypt it. Zuul serves each project’s public key using its build-in webserver. They can be fetched at the path /api/tenant/<tenant>/key/<project>.pub where <project> is the canonical name of a project and <tenant> is the name of a tenant with that project.

Zuul currently supports one encryption scheme, PKCS#1 with OAEP, which can not store secrets longer than the 3760 bits (derived from the key length of 4096 bits minus 336 bits of overhead). The padding used by this scheme ensures that someone examining the encrypted data can not determine the length of the plaintext version of the data, except to know that it is not longer than 3760 bits (or some multiple thereof).

In the config files themselves, Zuul uses an extensible method of specifying the encryption scheme used for a secret so that other schemes may be added later. To specify a secret, use the !encrypted/pkcs1-oaep YAML tag along with the base64 encoded value. For example:

- **secret:**

**name: test\_secret**

**data:**

**password: !encrypted/pkcs1-oaep |**

**BFhtdnm8uXx7kn79RFL/zJywmzLkT1GY78P3bOtp4WghUFWobkifSu7ZpaV4NeO0s71YUsi**

...

To support secrets longer than 3760 bits, the value after the encryption tag may be a list rather than a scalar. For example:

**- secret:**

**name: long\_secret**

**data:**

**password: !encrypted/pkcs1-oaep**

**- er1UXNOD3OqtsRJaP0Wvaqiqx0ZY2zzRt6V9vqIsRaz1R5C4/AEtIad/DERZHwk3Nk+KV**

**...**

**- HdWDS9lCBaBJnhMsm/O9tpzCq+GKRELpRzUwVgU5k822uBwhZemeSrUOLQ8hQ7q/vVHln**

**...**