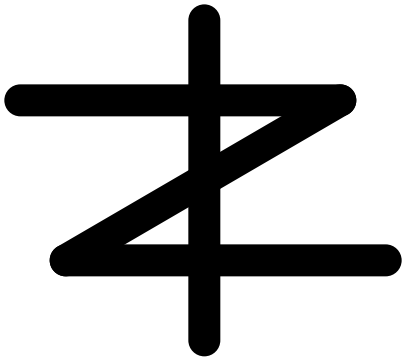
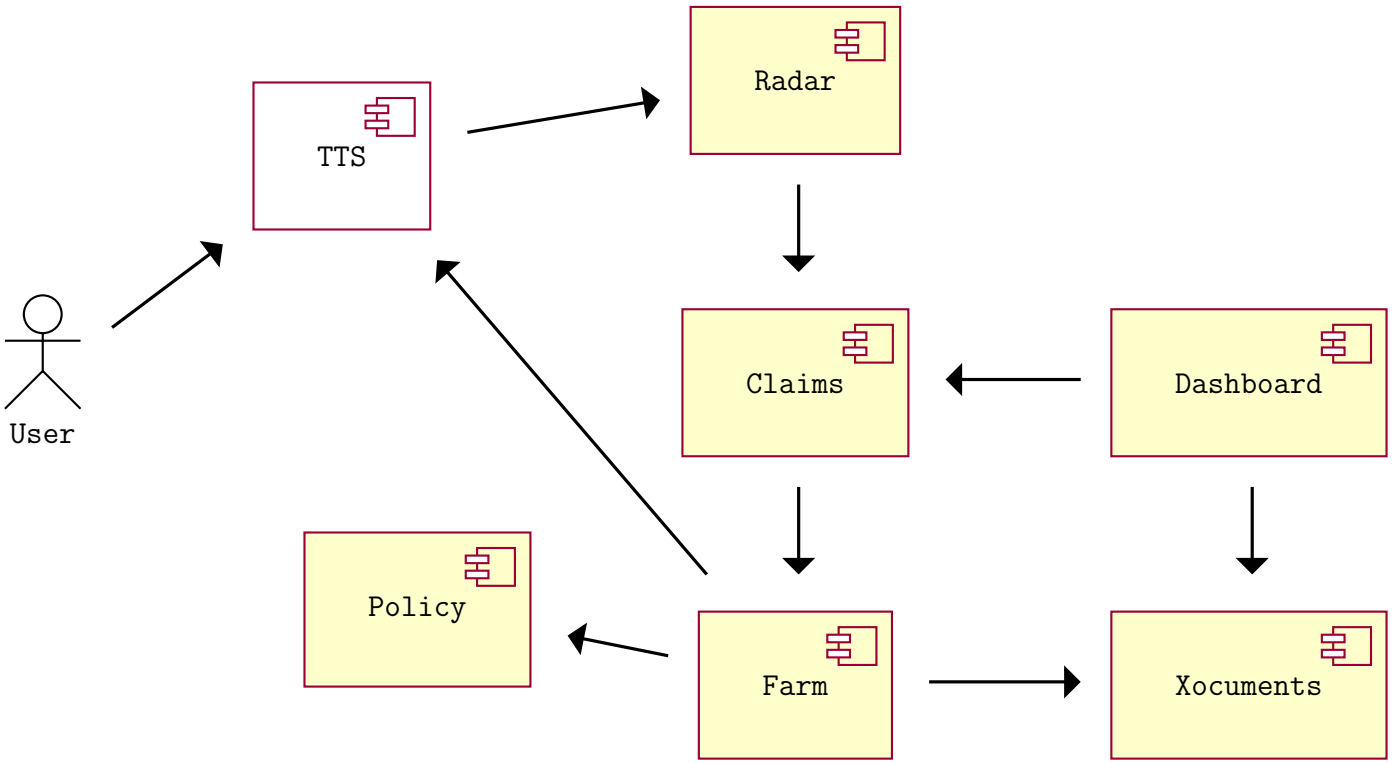


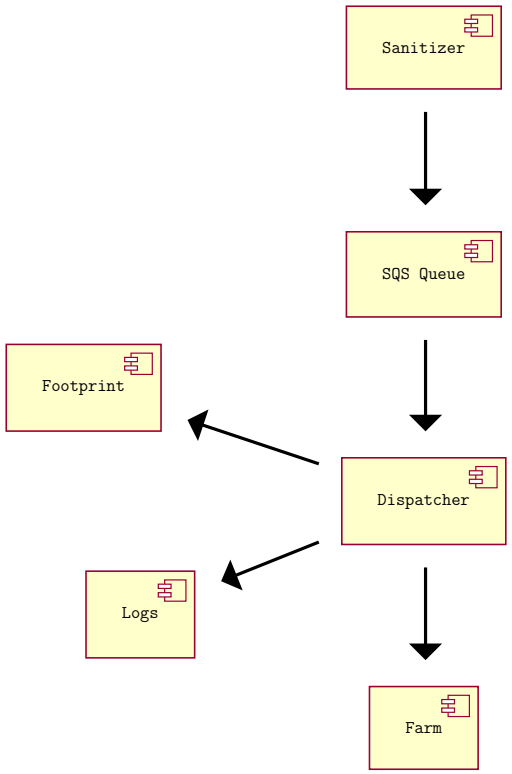
Zerocracy, Inc.

Zerocracy Architecture
www.zerocracy.com

Key Components



Claims



Java

The main programming language of the system is Java 8. The build system is Maven 3. The deployment is automated by Rultor. Continuous integration is done by Travis. Static analysis is performed by Qulice, an aggregator of Checkstyle and PMD.

SQS Queue

It is a large global cross-projects queue of claims, maintained in the AWS SQS. Anyone can drop a claim into the queue, it is a very fast operation. The processing of the claims happen later, when the dispatcher is up to it.

XML

Each claim is a short XML document with a pre-defined structure, validated against a XSD Schema. It includes information about the source of the event, the time and date, the type of request, and the list of supplementary arguments.

Footprint

There is a MongoDB database with a single table that records every claim that has been seen in the queue. Later, the footprint is accessible via the Dashboard, for monitoring and tracking purposes. The database is backed up every hour/day/week via ThreeCopies.

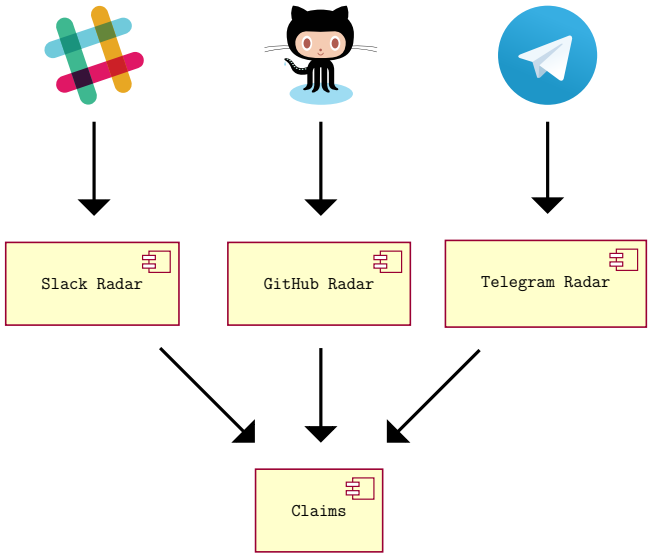
Cloud Logging

Each claim, after its processing via the Farm, produces logs, which are sent to the PaperTrail for monitoring purposes. On top of them, every exception is logged to Sentry.

XSL Sanitizer

Before a claim gets into the queue it is validated for its correctness by a series of XSL sanitizers. A claim may be rejected, for example, if it misspells the type or the login of the author.

Radar



Webhooks

Most radars receive events through pre-configured "webhooks," which send events in JSON format via RESTful API, which Zerocrat provides for them. For higher stability ReHTTP web service is used as a relay between chat platforms and radars.

DSL

Each chat platform speaks its own language, while claims must have specifically unified types. The domain specific language claims speak is understood by all Stakeholders in the Farm.

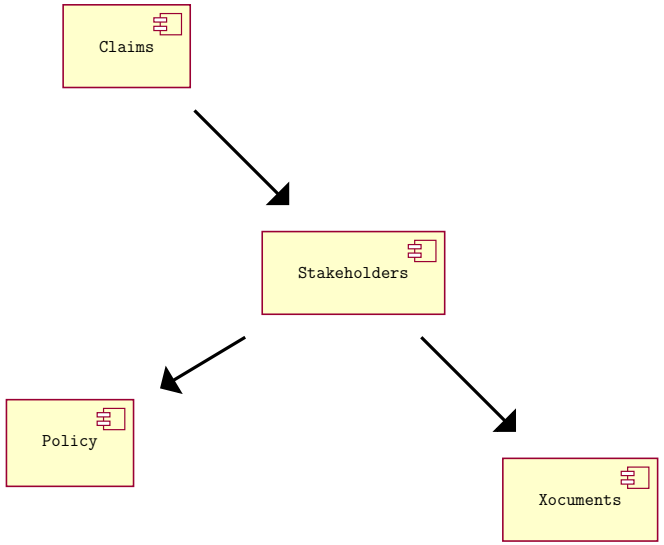
Decoupling

The way chat hubs are decoupled from Farm through the queue of claims gives a number of architectural benefits. First of all, the scalability is much higher, because the server doesn't need to reply immediately especially when the requests are coming in parallel. Second, longer response time is expectable by the user and the server can do many validating operations, no matter how complex is the request. The advantage of the chatbot architecture was explained in A Chatbot Is Better Than a UI for a Microservice blog post.

Modular Architecture

The number of external systems is growing and eventually Zerocrat will be integrated with a few dozens of them. Even though the complexity of each of them is not high, dealing with many of them at the same time is a serious challenge for the development team. Modular architecture simplified this task and makes the system more extendable.

Farm



Groovy

Each stakeholder is a simple procedure, usually 100-200 lines long, in Groovy. Each stakeholder is responsible for its own narrow and isolated operation, which is processing of a claim and submitting new claims to the queue. Stakeholders "talk" to each other only through the queue of claims.

Brigade

There is a collection of 150+ stakeholders (micro scripts), also known as "brigade," which perform individual management operations; the more complex is the Policy, the bigger the number of stakeholders.

Configuration

The behavior of stakeholders is configured via the Policy, an HTML document publicly hosted at zerocracy.com. The document is a mix of plain English text and placeholders for configuration parameters. Once a change is submitted to the Policy, the behavior of stakeholders change immediately.

XML

XML is the format of data for all project files (except the Ledger, which stays in the PostgreSQL database because of its very relational nature). The files are stored in AWS S3.

XSD Schema

Each XML document in the storage is validated against its corresponding XSD Schema and is rejected if there are any issues. Thanks to this validation all documents in the storage are always valid and correct.

XSLT

Each XML document in the storage has a corresponding XSLT stylesheet to convert it to HTML and render it in the dashboard.

Twitter Facebook Blog

Pitch Executive Summary Features

555 Bryant Str, Ste 470
Palo Alto, CA 94301
408.692.4742
team@zerocracy.com

0.14.0 January 5, 2019