OAK RIDGE National Laboratory

Employing DevOps in HPC Operational Management

By Aaron Barlow

ORNL is managed by UT-Battelle LLC for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

Introduction to Speaker & Organization

Aaron Barlow

HPC Software Engineer with over 8 years of industry experience

Oak Ridge Leadership Computing Facility (OLCF)

Provides world-leading high-performance computing (HPC) resources like Frontier to scientists and engineers to solve significant and complex problems in a broad range of scientific fields

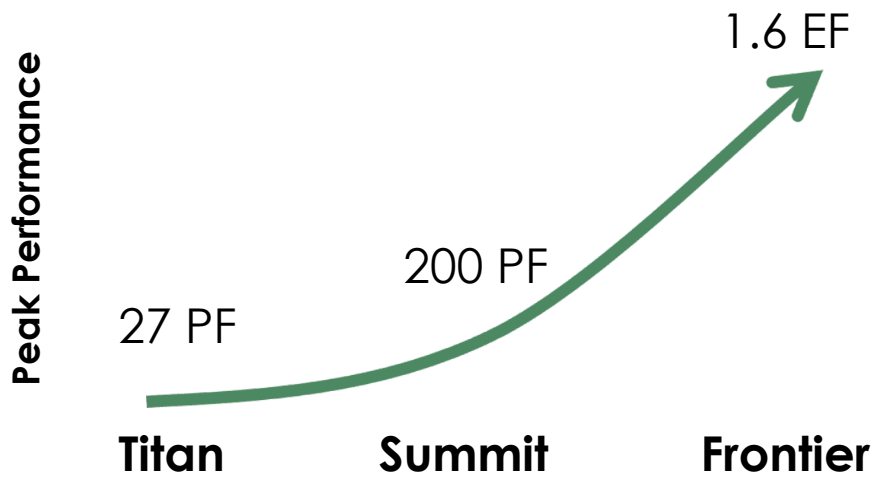
Software Services Development Group (SSD)

Develops and maintains large software applications and services used by the staff and end-users of the OLCF's computational ecosystem

HPC Operational Management

Scale and Complexity

- Our software supports the business of running HPC at our scale and complexity
  - Complexity: thousands of projects and users
  - Scale: Exaflop supercomputer Frontier



Provided Software Services

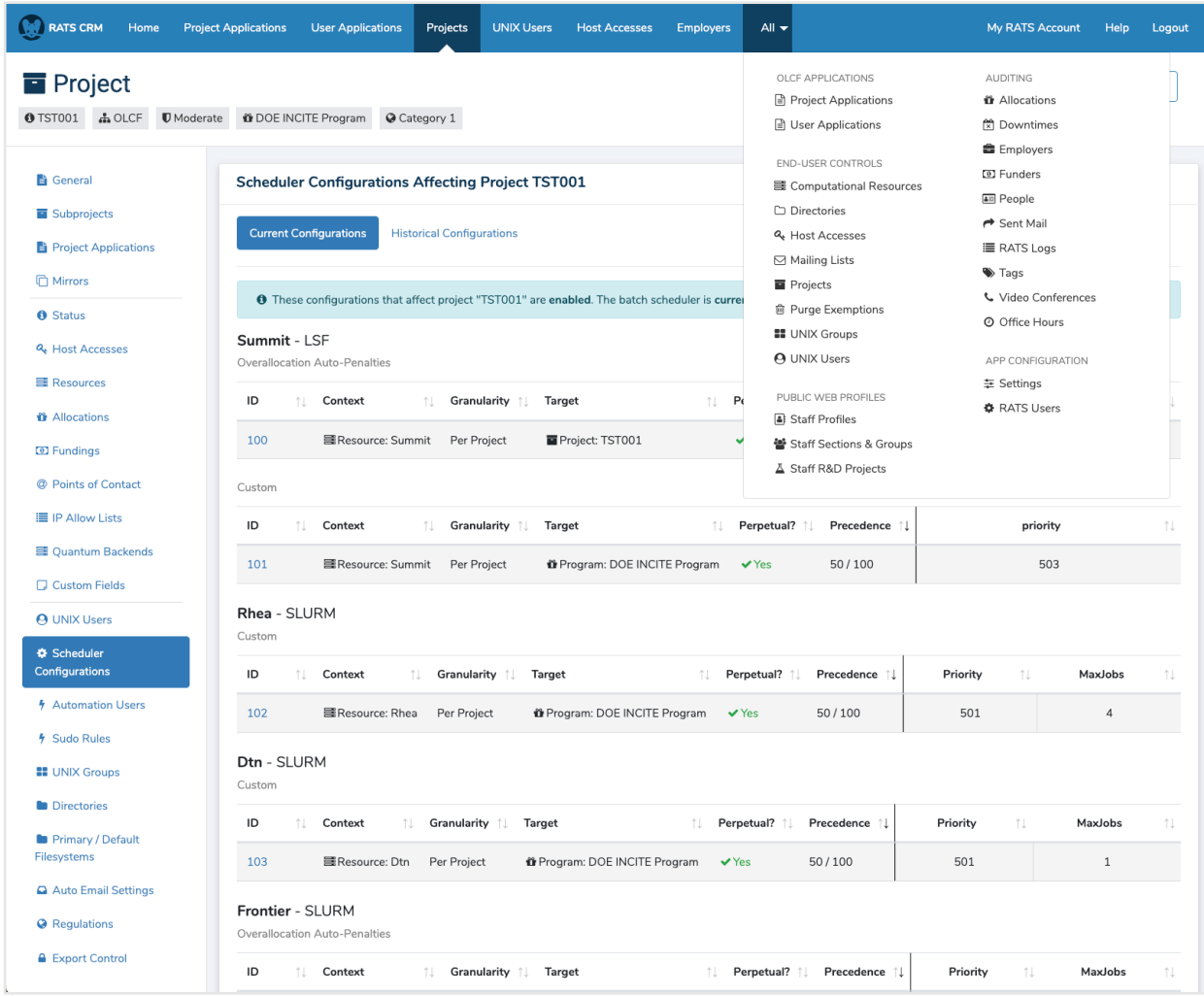
- User facing portal applications
- Business analytics systems
- Customer relationship management (CRM) applications
- API driven HPC system integration and management services
- Select auditing capabilities

Data provided by olcf.ornl.gov/frontier

RATS CRM: HPC Management Application

HPC Operational Management Software

- It directly controls many of the day-to-day operational needs of the center in real-time
- Some core HPC system features RATS CRM manages:
  - Filesystem directories
  - Scheduler configurations
  - HPC allocations
  - User and system access
  - Projects
  - Mailing lists
  - Auto sending select emails to users
- 162 API endpoints to disseminate information to our systems and services

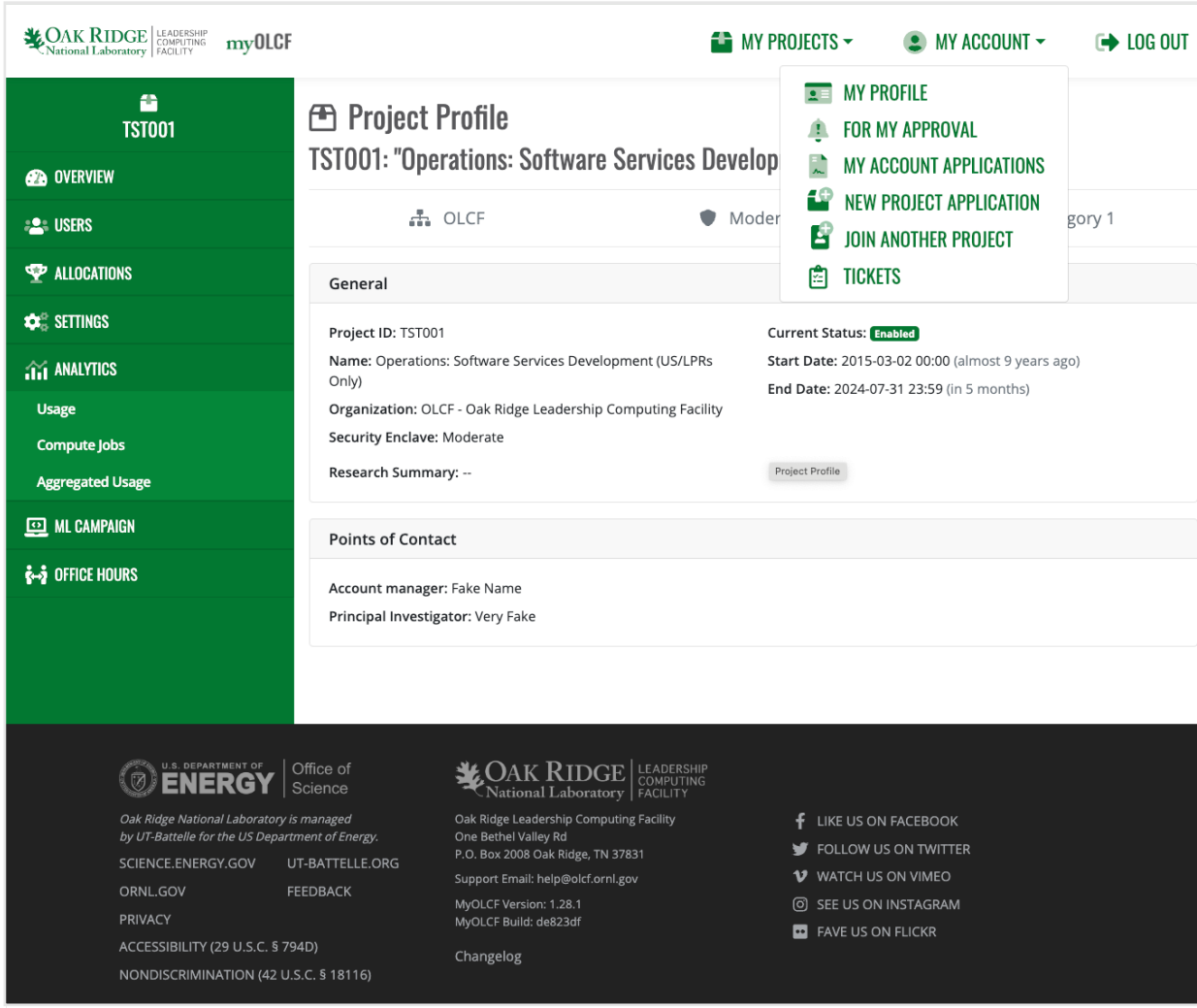


RATS-CRM with fake data

MyOLCF: Public Self-Service Application

Public HPC Self-Service Application

- Publicly available application that extends management capabilities and provides insights into available resources for users
  - my.olcf.ornl.gov
- Some key features:
  - Apply for and renew projects and accounts on HPC systems
  - Check the status of their applications
  - Monitor their submitted issues
  - Allocation usage monitoring
  - Compute usage report generation
  - Schedule office hours for help



MyOLCF GUI with fake data

Embracing DevOps and GitOps for HPC Management

Uptime

Reliability

Scalable

Quality

Security

Monitoring

Alerts

Reproducibility

Components of our GitOps

- GitLab Pipelines: automate our workflows
- Kustomize: configure Kubernetes applications & deployments
- Argo CD: providing continuous delivery of deployments
- Kubernetes: container management by Kustomize
- MinIO: object store for backups and caching
- KubeDB and Stash: database deployment and management
- SonarQube: identify bugs, vulnerabilities, and code quality issues
- HashiCrop Vault: securely store our passwords

GitLab Pipelines: Automating CI/CD Workflows

```
Running with gitlab-runner 16.6.1 (f5da3c5a)
  on gitlab-mini-runner-gitlab-runner-566c78b44-n2grn DKxTSZcr, system ID: r_UeulMNUpYQiW
Resolving secrets
Preparing the "kubernetes" executor
"ServiceAccount" overwritten with "gitlab-runner-builder"
Using Kubernetes namespace: stf040-ci-runner
Using Kubernetes executor with image image-registry.openshift-image-registry.svc:5000/stf040-ci-runner/slate-ci:latest ...
Using attach strategy to execute scripts...
Preparing environment
Using FF_USE_POD_ACTIVE_DEADLINE_SECONDS, the Pod activeDeadlineSeconds will be set to the job timeout: 24h0m0s...
Waiting for pod stf040-ci-runner/runner-dkxtszcr-project-949-concurrent-0-xhxbe8nn to be running, status is Pending
Waiting for pod stf040-ci-runner/runner-dkxtszcr-project-949-concurrent-0-xhxbe8nn to be running, status is Pending
        ContainersNotReady: "containers with unready status: [build helper]"
        ContainersNotReady: "containers with unready status: [build helper]"
Running on runner-dkxtszcr-project-949-concurrent-0-xhxbe8nn via gitlab-mini-runner-gitlab-runner-566c78b44-n2grn...
Getting source from Git repository
Fetching changes with git depth set to 1...
Initialized empty Git repository in /builds/uadev-team/rats-crm/.git/
Created fresh repository.
Checking out 8941a555 as detached HEAD (ref is refs/merge-requests/1816/head)...
Skipping Git submodules setup
Executing "step_script" stage of the job script
$ sha256=($(sha256sum Gemfile.lock | awk '{ print $1 }')) # collapsed multi-line command
$ export BUILD_ARG_BASE_VERSION=${BASE_VERSION}
$ export BUILD_ARG_NAMESPACE_UID=${NAMESPACE_UID}
$ if [ "${BUILD_DEBUG}" = "true" ] ; then # collapsed multi-line command
$ [ -z "${KUBE_NAMESPACE+x}" ] && KUBE_NAMESPACE=$(cat /run/secrets/kubernetes.io/serviceaccount/namespace) # collapsed multi-line command
$ oc create imagestream --namespace=${KUBE_NAMESPACE} ${IMG_STREAM} || true
Error from server (AlreadyExists): imagestreams.image.openshift.io "rats-crm-ci" already exists
$ BUILD_CHART=".build-templates/build-chart" # collapsed multi-line command
Cloning into '.build-templates'...
Using default SSD group build-chart for rats-crm-ci...
Set BuildArg: RUBY_VERSION=3.0.6
Set BuildArg: BASE_VERSION=c4bc76ef12003b0a6f00f704270042e481ed0df41c0a40f5a6e16101a86317a3
Set BuildArg: NODE_VERSION=16.18.1
Set BuildArg: NAMESPACE_UID=16647
Set BuildArg: NPM_VERSION=8.19.2
++ helm -n stf040-ci-runner upgrade rats-crm-2159618 .build-templates/build-chart -i --wait --reset-values --set BuildArgs.RUBY_VERSION=3.0.6 --se
   c76ef12003b0a6f00f704270042e481ed0df41c0a40f5a6e16101a86317a3 --set BuildArgs.NODE_VERSION=16.18.1 --set BuildArgs.NAMESPACE_UID=16647 --set Build
   t Name=rats-crm-ci --set Tag=8941a5552149a46557ffc287de6d2c630e7cd2a7 --set Dockerfile=container/ci/Dockerfile
Release "rats-crm-2159618" does not exist. Installing it now.
NAME: rats-crm-2159618
LAST DEPLOYED: Wed Feb 28 15:59:56 2024
NAMESPACE: stf040-ci-runner
STATUS: deployed
REVISION: 1
TEST SUITE: None
++ set +x
```

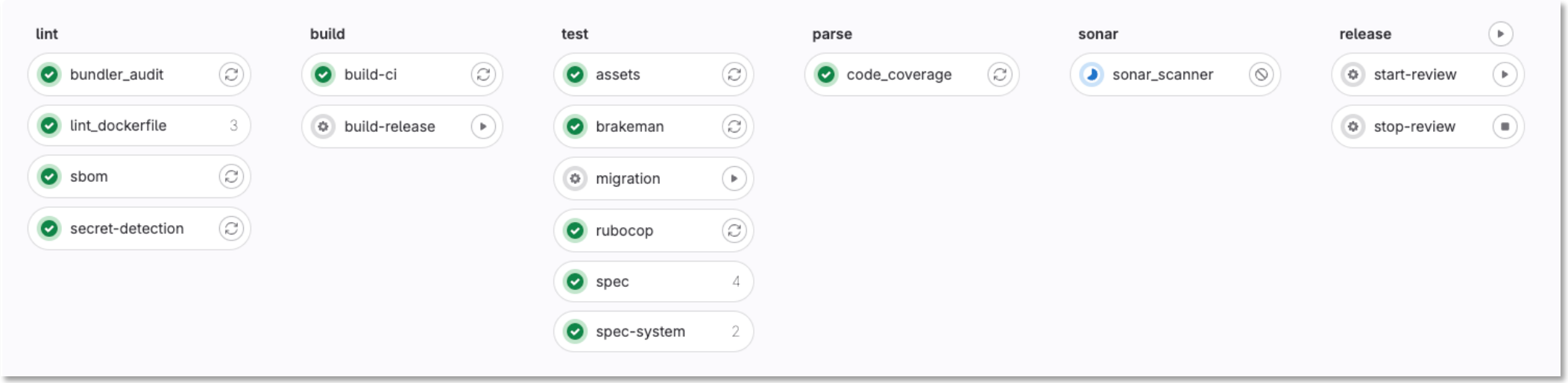
Real RATS-CRM build CI GitLab job

GitLab CI/CD Configuration

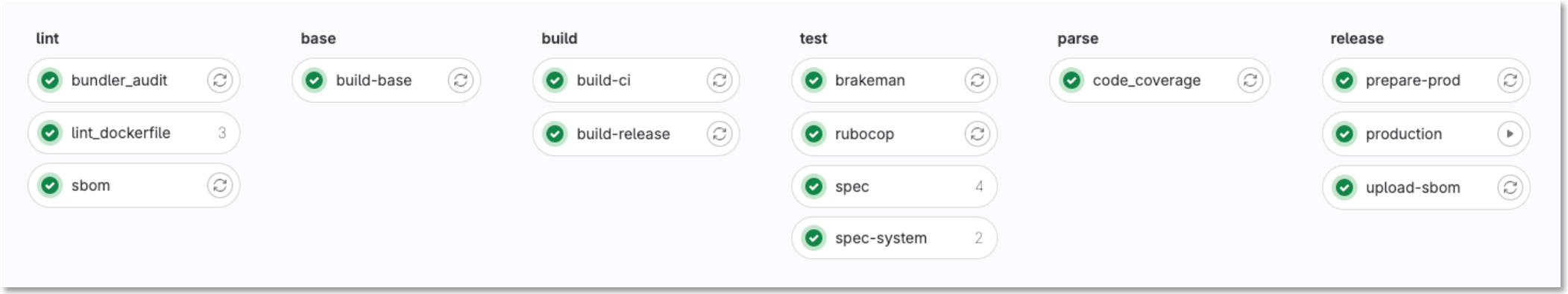
- GitLab jobs are run by GitLab Runners, which run on Kubernetes
- CI/CD workflows are defined in each repository's `gitlab-ci.yml`
- Key GitLab Pipeline features & benefits:
  - Automated and repeatable workflows
  - Efficiency with caching and parallel job execution

GitLab Pipelines: Visualizing CI/CD Workflows

Development Pipeline



Production Pipeline



Real RATS CRM GitLab Pipeline workflows

GitLab Pipeline Example: Code Coverage

Code Coverage Explanation

- GitLab Pipeline `code_coverage` step is defined in `gitlab-ci.yml`
  - `code_coverage` step defines process to generate and capture coverage metric(s)
  - `code_coverage` job generates a report job artifact which is observed by GitLab and made available

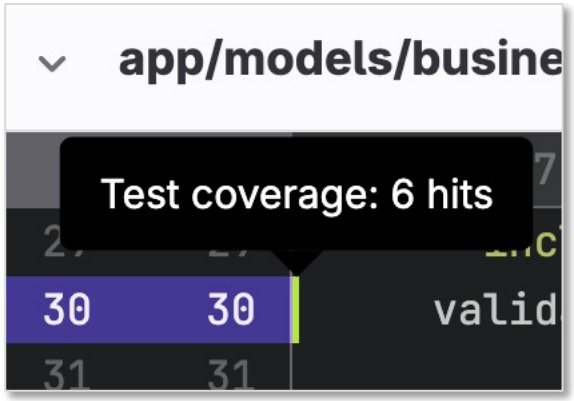
Code Coverage Job

```
30 Reading package lists...
31 Building dependency tree...
32 Reading state information...
33 The following additional packages will be installed:
34   xz-utils
35 Suggested packages:
36   seccomp
37 The following NEW packages will be installed:
38   glibc-source libseccomp-dev xz-utils
39 0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
40 Need to get 20.9 MB of archives.
41 After this operation, 27.6 MB of additional disk space will be used.
42 Get:1 http://deb.debian.org/debian bookworm/main amd64 xz-utils amd64 5.4.1-0.2 [471
43 Get:2 http://deb.debian.org/debian bookworm/main amd64 glibc-source all 2.36-9+deb12
44 Get:3 http://deb.debian.org/debian bookworm/main amd64 libseccomp-dev amd64 2.5.4-1+
45 debconf: delaying package configuration, since apt-utils is not installed
46 Fetched 20.9 MB in 0s (88.5 MB/s)
47 Selecting previously unselected package xz-utils.
48 (Reading database ... 15610 files and directories currently installed.)
49 Preparing to unpack .../xz-utils_5.4.1-0.2_amd64.deb ...
50 Unpacking xz-utils (5.4.1-0.2) ...
51 Selecting previously unselected package glibc-source.
52 Preparing to unpack .../glibc-source_2.36-9+deb12u4_all.deb ...
53 Unpacking glibc-source (2.36-9+deb12u4) ...
54 Selecting previously unselected package libseccomp-dev:amd64.
55 Preparing to unpack .../libseccomp-dev_2.5.4-1+b3_amd64.deb ...
56 Unpacking libseccomp-dev:amd64 (2.5.4-1+b3) ...
57 Setting up libseccomp-dev:amd64 (2.5.4-1+b3) ...
58 Setting up glibc-source (2.36-9+deb12u4) ...
59 Setting up xz-utils (5.4.1-0.2) ...
60 update-alternatives: using /usr/bin/xz to provide /usr/bin/lzma (lzma) in auto mode
61 $ curl -sfL https://raw.githubusercontent.com/golangci/golangci-lint/master/install.
62 golangci/golangci-lint info checking GitHub for tag 'v1.55.2'
63 golangci/golangci-lint info found version: 1.55.2 for v1.55.2/linux/amd64
64 golangci/golangci-lint info installed /go/bin/golangci-lint
65 $ make test-quality
66 golangci-lint --version
67 golangci-lint run --timeout="2m0s"
68 golangci-lint has version 1.55.2 built with go1.21.3 from e3c2265f on 2023-11-03T12:
69 Cleaning up project directory and file based variables
70 Job succeeded
```

GitLab Test Coverage

A Merge Request's test coverage

Individual code changes' test coverage



Note: all examples are real, and the test coverage shown is for Jacamar CI (a public repository)

Kustomize: Streamlining Kubernetes Configurations

Kubernetes Configuration

- Open-source Kubernetes configuration management
  - We use Kustomize to configure our Kubernetes deployments
- Key Benefits:
  - Define base configurations and use overlay configurations apply patches
  - Uses configuration inheritance to minimize redundant configuration
  - Easy integration with Kubernetes (kubectl)
  - Open-source

```yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

resources:
  - deployment-app.yaml
  - service.yaml
  - serviceaccount.yaml
  - stateful-sphinx.yaml

configMapGenerator:
  - name: nginx
    files:
      - configs/nginx/nginx.conf
      - configs/nginx/security.conf
      - configs/nginx/listen.conf
  - name: rats-cfgs
    files:
      - configs/rats/ldap.yml
  - name: sphinx-svc
    files:
      - configs/rats/thinking_sphinx.service.yml
  - name: rats-scripts
    files:
      - scripts/migrate_index.bash
      - scripts/prep_server.bash

secretGenerator:
  - name: rats-env
    files:
      - configs/rats/database.yml
      - configs/rats/secrets.yml
configurations:
  - kustomizeconfig.yaml

generatorOptions:
  labels:
    app.kubernetes.io/component: configuration
```

Real Kustomize configuration example

Kustomize vs Helm: Key Differences

Kustomize

1. No Templates
2. Applies changes with patches and overlays

Pros

- Template free
- Configuration simplicity

Cons

- Fewer features

Helm

1. Uses templates
2. Uses values file to adjust settings in template

Pros

- Vast repository of preconfigured deployments
- More popular with 2.5x the stars on GitHub

Cons

- Steep learning curve

Why Migrate to Kustomize?

1. Strong control over Kubernetes ConfigMaps
2. We don't support many deployments
3. Attaches checksum to configurations to ensure configuration change
4. Simplified configuration

Harnessing the Power of GitOps with Argo CD

What is GitOps?

- Git driven operations
- Uses Git to define infrastructure
- Automates deployment and enables infrastructure version control

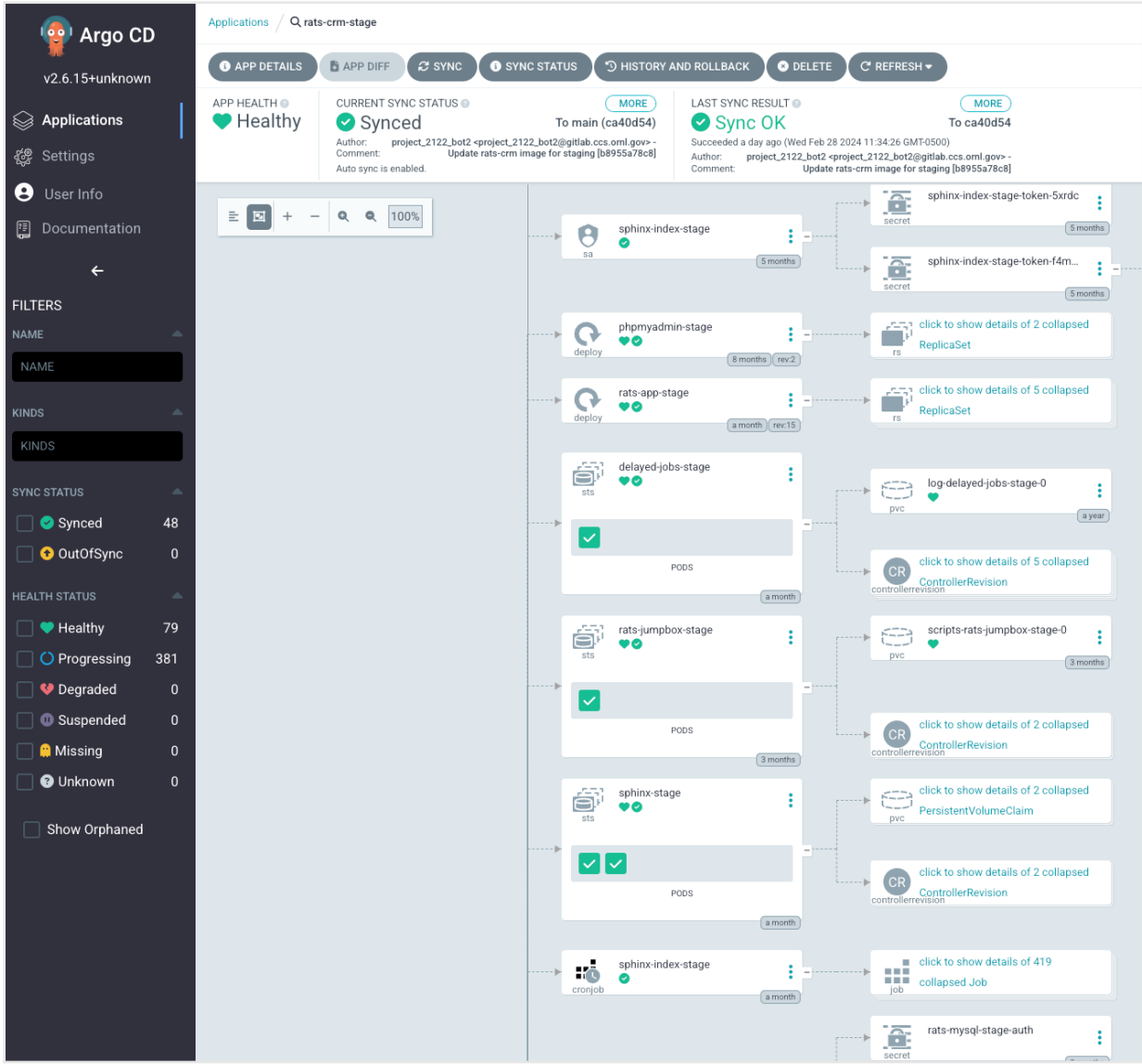
What is Argo CD?

- Our GitOps deployment tool
- It's a declarative continuous deployment for Kubernetes

Argo CD: Automated Kubernetes Deployment

Argo CD Key Features

- Automatically syncs our applications to ensure desired state defined in our Kustomize configurations
- Provides a GUI for real-time monitoring and management of applications
- Provides us the ability to rollback and see our changes
- Automatically fixes drift

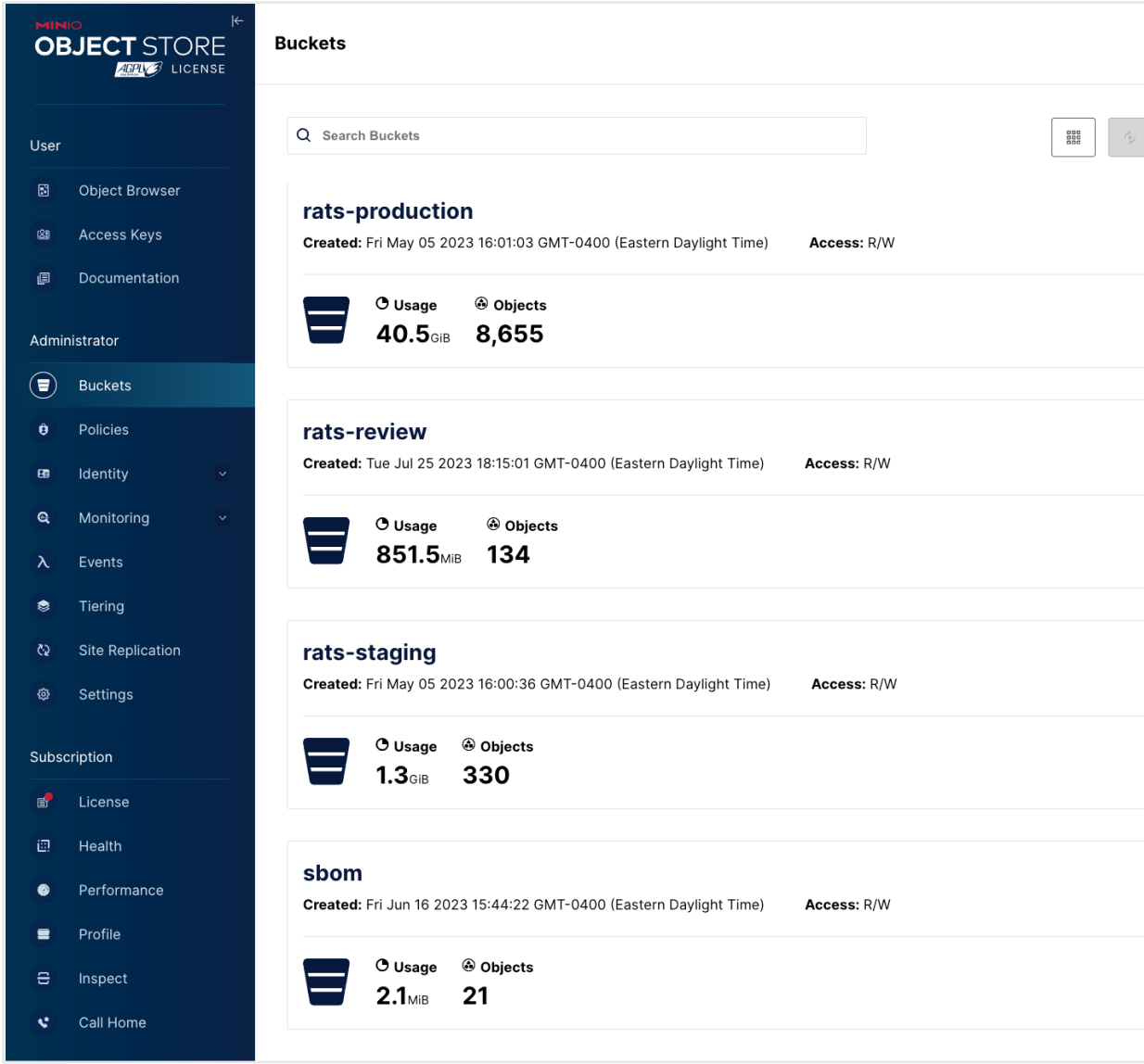


Argo CD: real partial view of RATS CRM

MinIO: Scalable Cloud Storage

Cloud-native Object Store

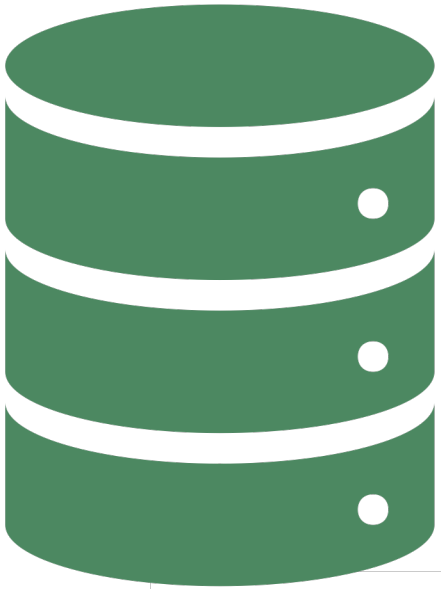
- Offers scalable object storage with high performance and reliability
- We use MinIO for
  - Database backups
  - GitLab Runner cache
  - User-uploaded file storage
  - User-uploaded file backups
  - Internal application backups
  - Tracking Software Bills of Materials (SBOMs) for each release



Real MinIO instance

KubeDB and Stash: Database Management

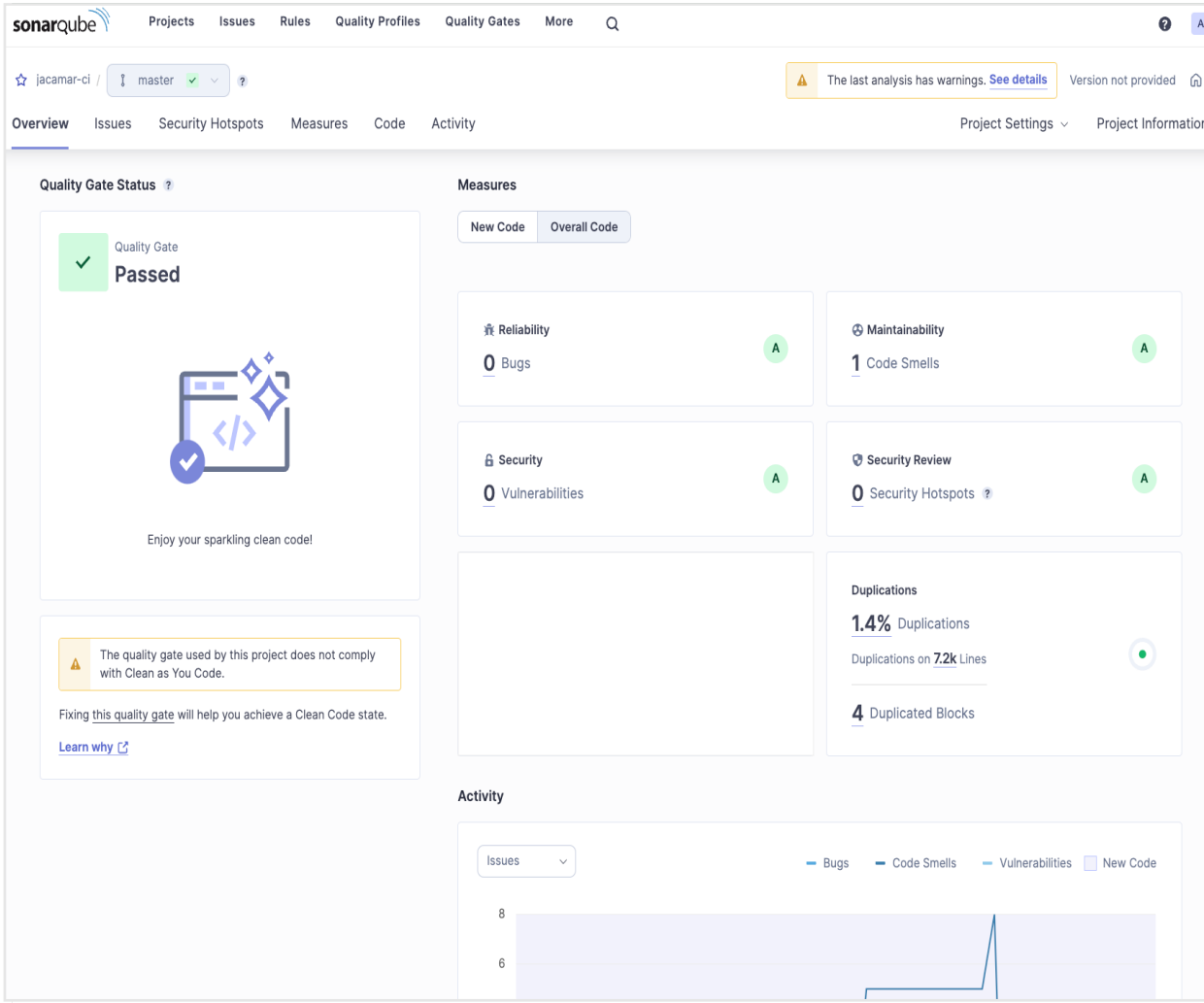
- KubeDB
  - Administrates the complete workflow for deploying and managing our databases on Kubernetes
- Stash
  - Provides us an easy data backup and recovery on Kubernetes
  - We use Stash in our review environment with heavily obfuscated data that we restore to our review deployment's database
- We integrate these services with Kustomize



SonarQube: Application Quality Assurance

Cloud-native Quality Assurance

- Integrated into our development GitLab pipelines
- Provides automated:
  - Code quality ("smells") based on preset rules
  - Security vulnerability detection
- Provides timeline view of our project's quality in several domains



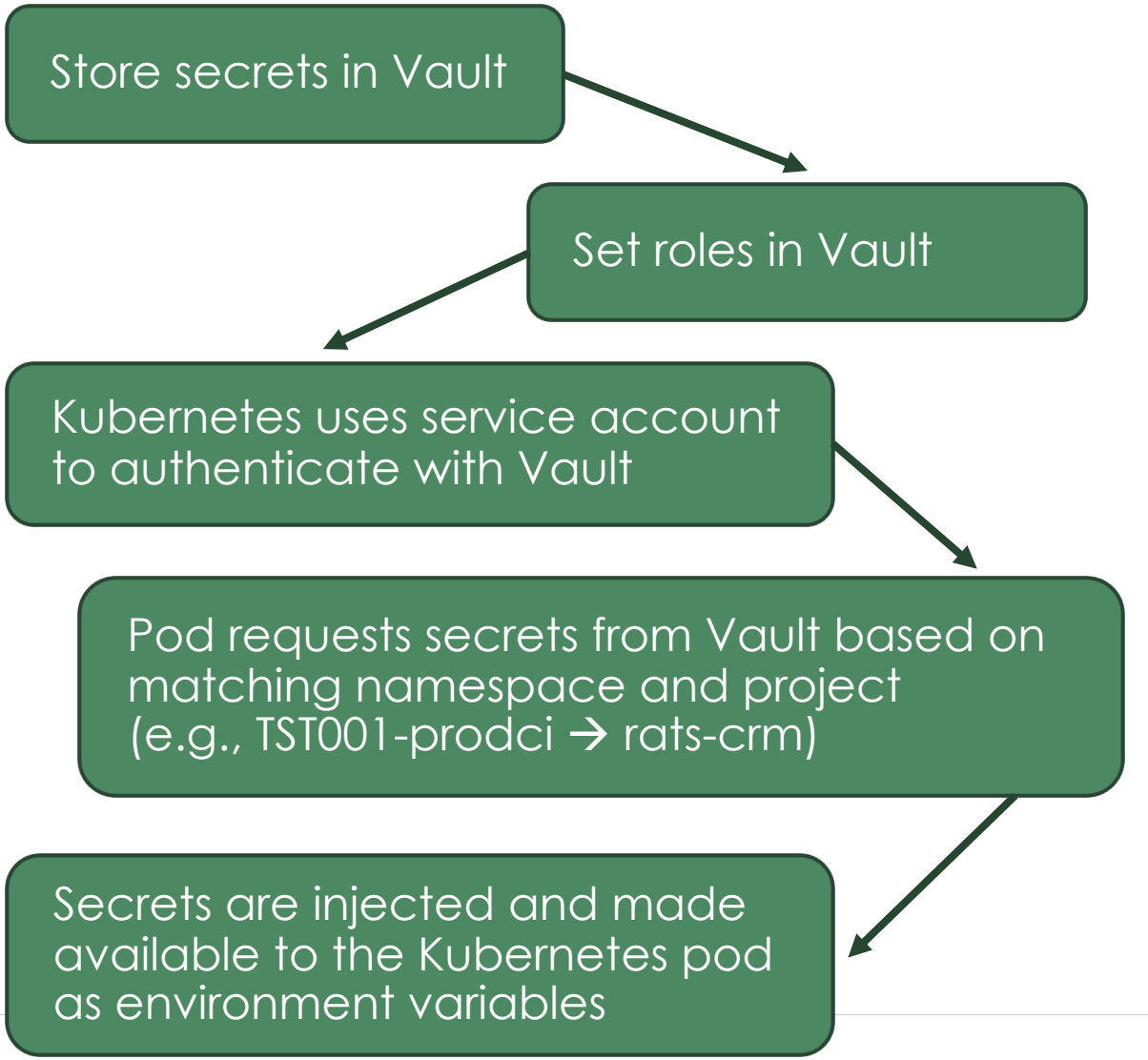
View of Jacamar CI on real SonarQube instance

Note: the SonarQube data is from a public repository

HashiCorp Vault: Securing Digital Assets

Cloud-native Secure Secrets Store

- Open-source software for secure secret storage with versioned secrets and access management
- Our secrets are synced to our Kubernetes secrets
- Our software applications can access the secrets as environment variables



OLCF Acknowledgement

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

OAK RIDGE National Laboratory

Questions