coveros

COVEROS IMPLEMENTS CUTTING-EDGE DEVSECOPS SOLUTION

CASE STUDY

Coveros Implements Cutting-Edge DevSecOps Solution

Coveros recently helped an industry-recognized healthcare solutions company advance its mission of optimizing human health through precision technology and digital tools for preventing, monitoring, diagnosing, and treating diseases. This organization engaged Coveros to build an industrial-strength DevSecOps platform and pipeline as well as automating testing to assure the quality of production software.

CHALLENGES

- The organization's product is built using an in-house, rule-engine-based platform, which requires build, deployment, and testing for both the software application code as well as the program content configuration.
- The organization has an unpredictable release process and a day-to-day firefighting mentality.
- The company's current test process, code coverage, and test data management does not result in a quality product in production.
- The product has significant data privacy and security requirements that have not been fully satisfied.
- There is little to no automated testing to support accelerating delivery.
- The existing cloud-based Virtual Machine (VM) infrastructure is hand-crafted and has limited ability to scale. This makes it challenging to meet changing production volume or provide the proper monitoring necessary to detect and recover from problems.
- The organization found that converting from VM-based hosting to container-centric Kubernetes was more difficult than expected.
- Stability problems with the organization's Kubernetes platform in the development environment led to confidence problems in moving to Kubernetes as a production platform.
- The duality of Kubernetes-for-development and VM-for-prod is causing significant defect rates in production.

The organization began the journey of moving from Virtual Machine (VM)-based deployment to a container-based model using Kubernetes prior to engaging Coveros. Unfortunately, their set of Virtual Machines in Amazon Web Services evolved over time into highly customized and sometimes fragile environments, leading to difficulties deploying software and configuration updates into the various development, test, and production environments. While much of the software deployment process was automated, the implementation of the Ansible scripts for deployment had grown complicated and difficult to maintain.

The organization also lacked a strong test automation approach integrated into its DevSecOps pipelines to address ongoing quality and security issues. Specifically, they lacked the automated tests necessary to enforce the quality of new features, as well as the regression tests needed to ensure that older features remained functional when changes were introduced. Application security assurance was not part of the end-to-end DevOps process resulting in significant late lifecycle rework. The organization's rule-engine-based platform hosts content-agnostic programs supported by three device platforms: Web, Android, and iOS. The enormous interoperability scope of this setup made it nearly impossible to fully cover features and end programs

Coveros Implements Cutting-Edge DevSecOps Solution

with traditional manual testing. Additionally, the organization had a QA team with limited technical abilities, making it difficult to create maintainable test automation.

SOLUTION

Coveros led an effort to architect, design, implement, and test a scalable, secure, highly-available, self-healing enterprise system that would incrementally replace the existing fragile, static delivery and production environments. We also created and led the establishment of a test automation capability within QA and integrated better regression, functional testing, and security testing into the development and delivery process over time. A key component of this approach was the development of a test automation framework that eases the process of creating and maintaining automated quality and security tests.

Initially, the Coveros team leveraged our industry-leading DevSecOps Maturity Model™ and Test Process Improvement Model™ to quickly assess the current maturity of continuous integration, continuous delivery, test automation, and application security of the organization's existing delivery and production platforms and practices. We quickly identified key gaps in the existing DevSecOps process, Kubernetes approach, and testing efforts. Based on this analysis, we created an improvement plan to build a delivery and production solution that effectively integrated test automation into the process. Two Coveros teams were deployed to implement our recommendations.

DEVSECOPS PLATFORM AND PIPELINE IMPLEMENTATION

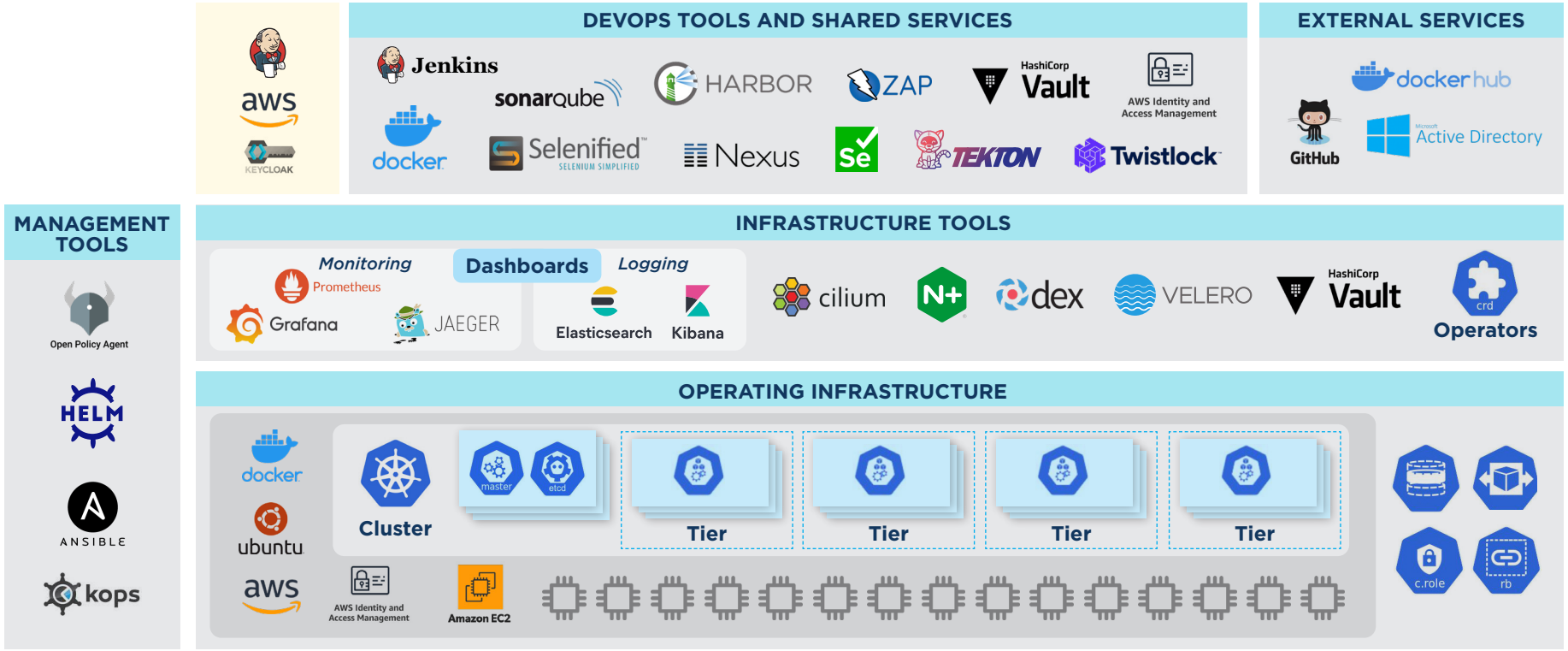
In order to properly support the goals of a highly-available, self-healing enterprise system, the Coveros team designed, built, and automated a scalable architecture leveraging Kubernetes clusters that enhance the organization's core software product and platform. This involved design and implementation of:

- An architecture for the Kubernetes operating platform along with the automation of the installation, configuration, and maintenance of all the infrastructure tools necessary to support it.
- Design and implementation of a CI/CD system to build, test, secure, and deploy updates to the organization's healthcare product, supporting configuration content, and underlying infrastructure with an integrated set of quality and security assessment gates.
- A set of established dashboards and other information radiators to easily view the health and performance of the entire system.

We deployed a broad set of tools to provide a highly-reliable and secure continuous integration and delivery process for the organization's key products and platform.

The Coveros team used an "Infrastructure as Code" approach to capturing all configurations using a library of Ansible deployment scripts to support the organization's broad set of tools and complex infrastructure. This enabled the team to create an entire Kubernetes cluster in less than 30 minutes where software could be deployed and tested. It also allowed the team to roll out platform updates to a half

Coveros Implements Cutting-Edge DevSecOps Solution

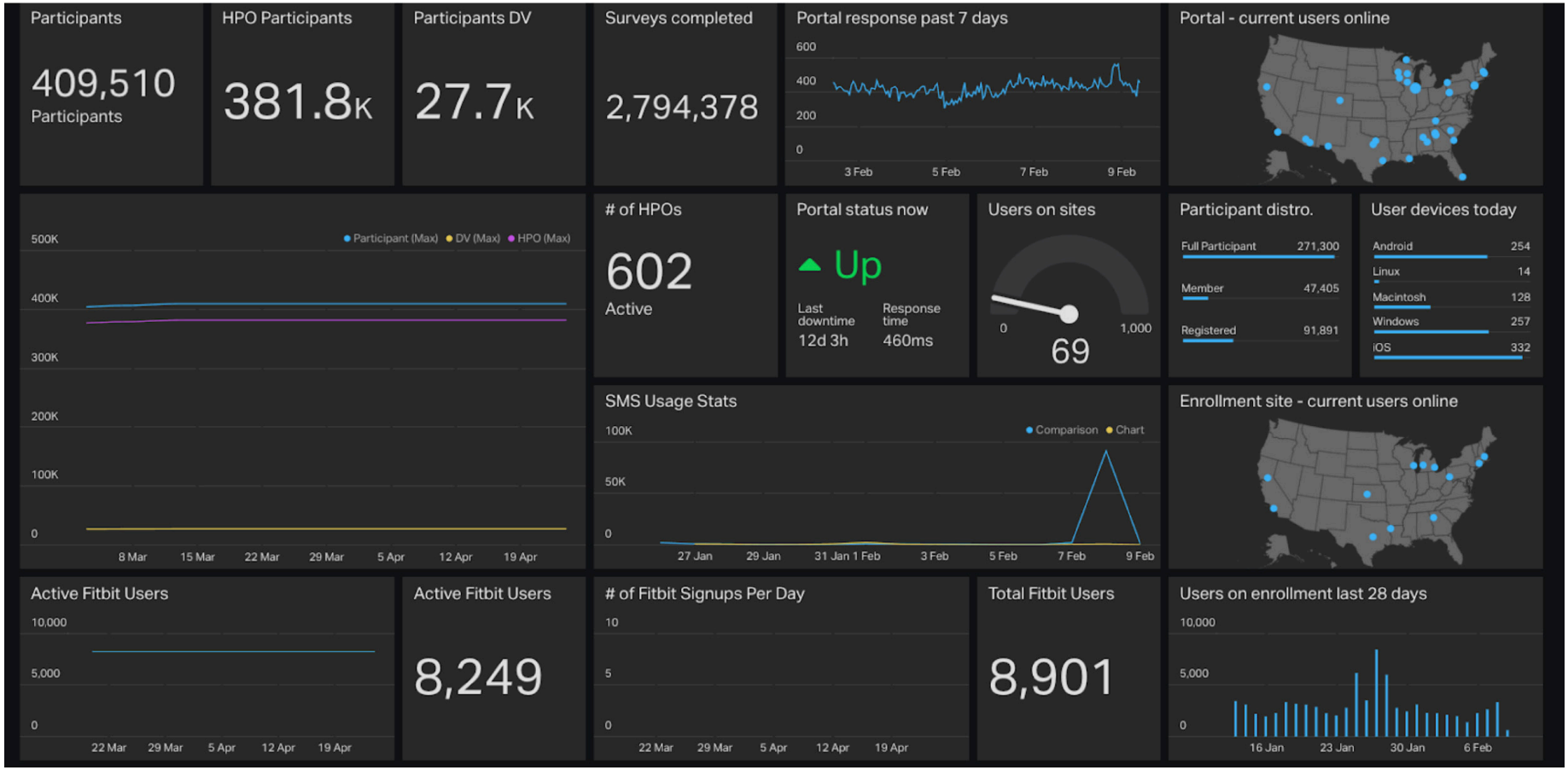


dozen test and production clusters with 200-300 AWS EC2 nodes using a GitOps approach and an automated Tekton pipeline.

For monitoring, alerting, and dashboards, the Coveros team deployed tools to gather metrics, aggregate logs, and provide distributed application tracing data. We used Prometheus and Grafana for performance and resource monitoring. We used ElasticSearch and Kibana for log aggregation and log analysis. We built integrated alerting with Prometheus Alert Manager and ElastAlert to detect problems and notify operational or security staff using email, Slack, and PagerDuty. We established a set of dashboards using Grafana and Kibana to constantly monitor and diagnose resource utilization, performance metrics, and error rates within the system.

Coveros Implements Cutting-Edge DevSecOps Solution

As part of the DevSecOps Automation effort, the Coveros team implemented a full CI/CD pipeline that integrated all the proper build, scan, package, test, secure, and deploy steps necessary to move the software from source code to operating software in production. The tech stack includes build automation using a "Pipeline as Code" approach with Jenkinsfiles linked into all the source repositories. The CI build compiled



the code, ran all the unit tests, and scanned the code for security and quality defects using SonarQube to establish the pipeline-produced deployable Docker containers. These containers were then automatically installed into on-demand dynamic test environments where all the automated quality and security tests were executed. Static application security testing (SAST), dynamic application security testing (DAST), and software component analysis (SCA) were integrated throughout the delivery process to verify the security of the application continuously. We used Open Policy Agent (OPA) to define deployment policies and Conftest to validate OPA policies during the delivery process. Twistlock was used for container scanning and monitoring. Upon successful results, the software was then automatically deployed into permanent QA and Staging environments. Production development used the same deployment automation to deploy the software updates as an on-demand business decision by the release managers.

TEST STRATEGY AND AUTOMATION

To address existing quality problems identified during our assessment process, Coveros created a test automation framework that would allow the organization to integrate test automation directly into the CI/CD process. The framework used Cucumber, a BDD framework, to test user scenarios across multiple device platforms. We wrote API tests using Cucumber to test the web services provided by the application. By utilizing a BDD testing framework, the Quality Assurance team was empowered to write automated tests using business language instead of code. We used Appium and Selenium to allow the tests to execute on virtual and physical iOS and Android devices as well as the most popular web browsers, such as Google Chrome, Mozilla Firefox, and Microsoft Edge. We leveraged our open-source test automation framework, Selenified, as a base framework upon which the tests relied for reporting and wrapping WebDriver calls.

To make writing automated tests easier for non-technical personnel, Coveros stood up an instance of our open-source BDD writer, GherkinBuilder, to provide an easy-to-use web interface for anyone writing Gherkin. We configured the tool to pull in the test steps for the platform and program, to allow auto-completion of existing tags and test steps, as well as integrating with the organization's build tool and test management tool. With these tools, manual testers or business analysts could write, execute, and view tests through Jira, without the need for a cumbersome development environment setup.

Coveros created a test management solution using the Zephyr for Jira plugin to manage tests and record test executions. In order to keep a single source of truth, we wrote the tests themselves as Gherkin scenarios and stored them in feature files, which were contained within the project's source control repository. Each test was tagged with its Jira ticket number to keep tests synchronized. Every time a test is executed by the framework, the framework uses the Jira and Zephyr APIs to ensure all test information is copied from source control to Jira. After test execution, the framework uploads all test results to Zephyr, creating easily-viewable traceability, history, and run reports.

In order to manage the growing codebase of the testing framework and automated tests, the Coveros team implemented a continuous in-

Coveros Implements Cutting-Edge DevSecOps Solution

tegration pipeline. Every push to source control (Git) and pull request opened had this CI run against the new changes introduced. We analyzed all code and tests (Java and Gherkin) against coding best practices from SonarQube, to ensure that no issues were introduced. Additionally, we ran unit and integration tests to ensure basic functionality worked as desired. Finally, by utilizing Docker and Kubernetes, a sample application was spun up, and a subset of tests was run against it, to ensure they still worked. Only builds that passed all checks were allowed to be merged into the main testing codebase. Once a build did pass, an artifact of the testing framework or workflow was pushed to a Docker repository (Nexus), so that tests could be easily downloaded and run by any individual.

THE OUTCOME

The CI/CD pipeline enabled the team to rapidly build and deploy hundreds of small code changes into the various test, staging, and production environments across the enterprise. The integrated testing and security assessment gates prevented hundreds of builds containing defects that normally could have escaped into customers' hands. The Kubernetes platform provided stability that allowed the VRP system to react to surges in customer activity, detect problems in the system, and self-heal when problems were discovered. Throughout the implementation, the development and leadership teams were able to monitor and track the health of the system using integrated dashboards and metrics.

The organization was able to move from having no standard automated tests to over 1,700 tests that verified their application. Test automation was also leveraged to validate the application on more than 200 supported OS/device combinations. Both technical and non-technical teams are empowered to write tests, allowing the automation team to focus more on triage, upkeep, and new features. This has increased confidence in production releases and improved the efficiency of system and smoke tests to verify that the platform and program were successfully deployed.

All these benefits have allowed the organization to successfully launch its platform nationwide, and continue to release updates and enhancements more quickly and with confidence in product quality. Moving to more elastic environments also allowed the organization to reduce their cloud costs by $500,000 per month.