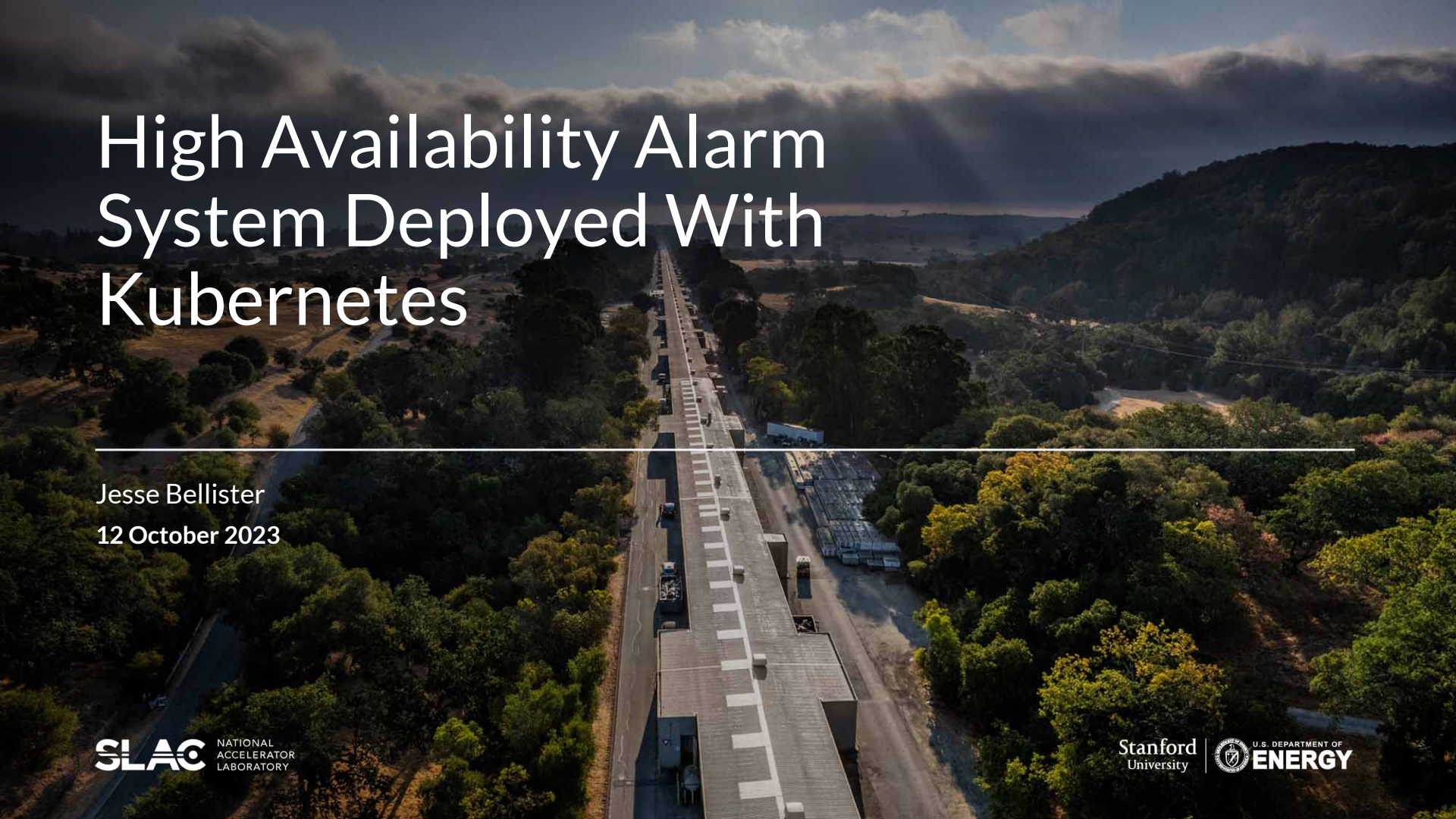
High Availability Alarm System Deployed With Kubernetes

Jesse Bellister

12 October 2023



SLAC NATIONAL ACCELERATOR LABORATORY

Stanford University | U.S. DEPARTMENT OF ENERGY

Introduction

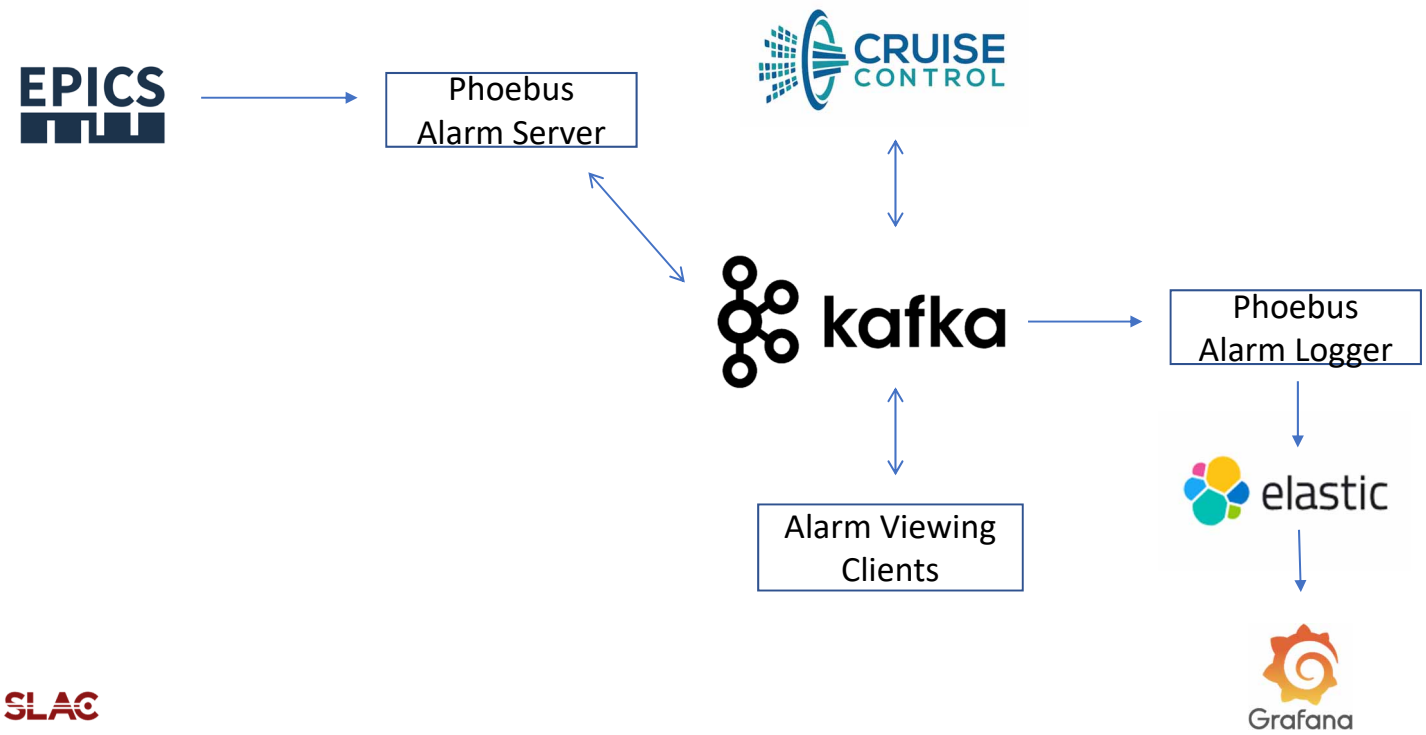
Main Goals for the Project

- Increased operator engagement with the alarm system
- Easy to keep the system updated as devices are added or removed
- Robust deployment with near constant uptime
- Keep in line with current best practices for technologies used

Stakeholders

- Accelerator Operators, Scientists, End-Users
- IOC Engineers
- System Administrators

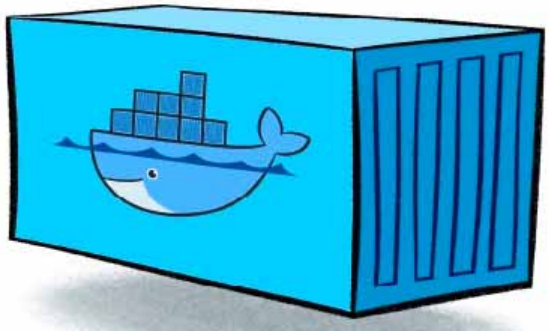
Phoebus Alarm Server



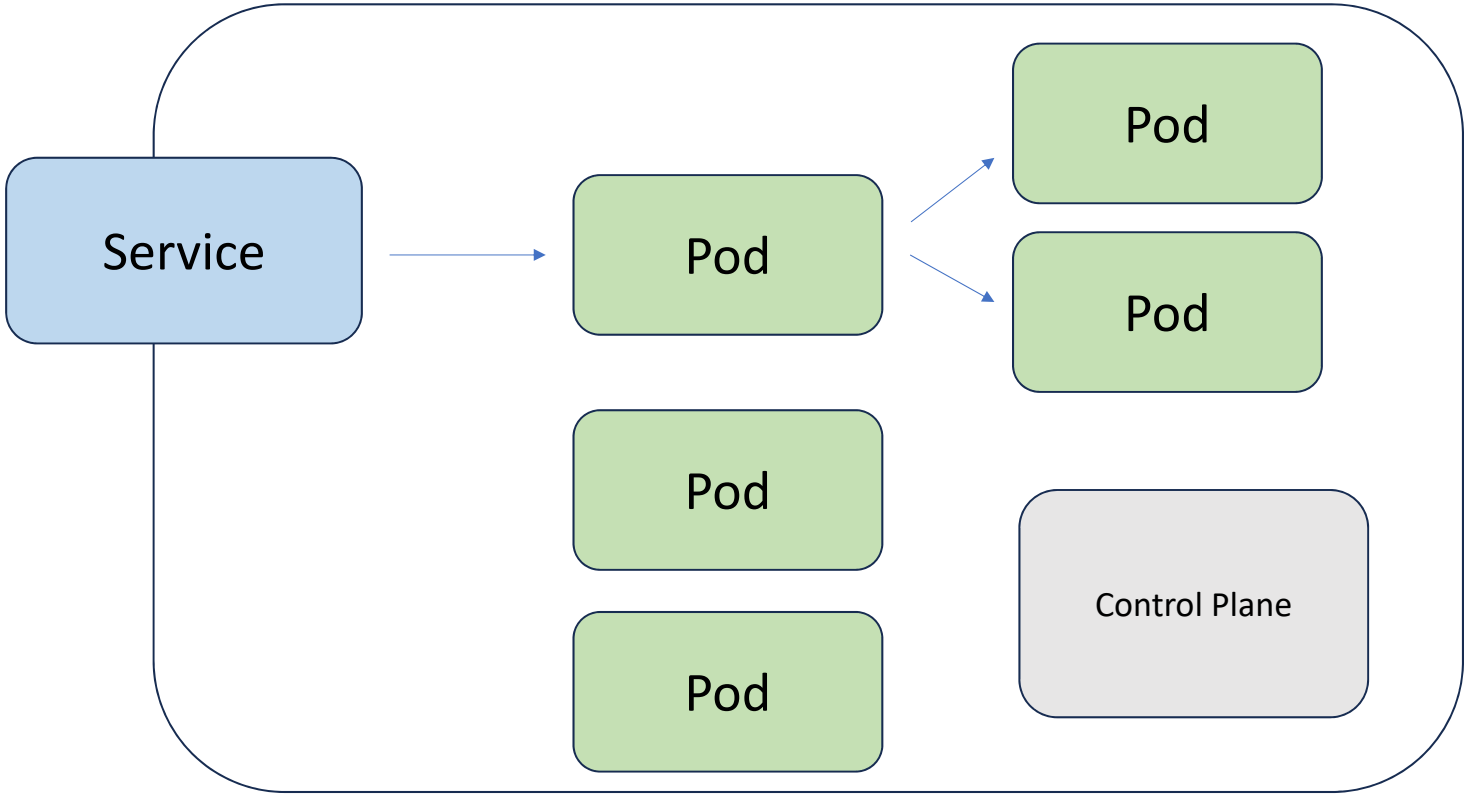
Containerization

Why?

- All dependencies are easy to track
- Build anywhere
  - Don’t like yum? Can use apt instead even if the host OS is CentOS
- Consistent environment
  - Self-documenting, file for generating container is version controlled with git
  - Shareable
- Reasonable learning curve
- But how do we want to deploy and manage these containers?



Kubernetes



Kubernetes

What are the benefits?

- High Availability
  - Kubernetes control plane is constantly monitoring all components of the deployment
  - If a node goes down, pods running on it are automatically moved to a different node
  - Set a number of replicas for each component

- Declarative Configuration
  - Tell the cluster what you want, not how to do it
  - “Give me 2 alarm servers and 3 kafka brokers, at least one of each must always be running”

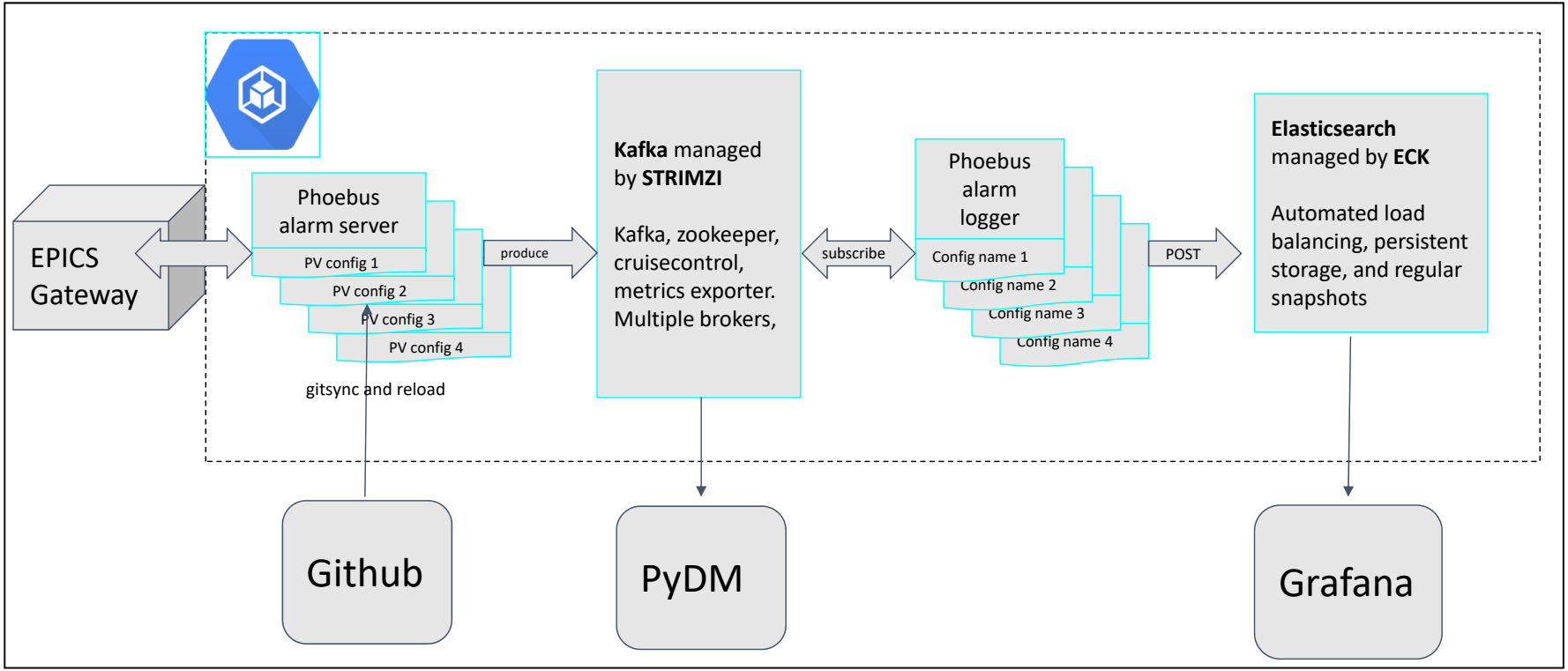
- Deployments are easily replicated
  - Manifest files are defined with yaml, version controlled with git
  - Deployment of entire system can be done with a one line command “kubectl apply –k base”

Alarm System Deployment

Focus on High Availability and Ease of Upgrade

- Kubernetes cluster is running in the SLAC Shared Scientific Data Facility (S3DF)
  - Virtual cluster (vcluster) isolates our system from everything else in the cluster
- Multiple replicas of each Phoebus alarm server are running
  - Pod anti-affinity ensures each copy runs on a separate node
  - Pod disruption budget ensures at least one copy is always running
  - Set a number of replicas for each component
- Operators are used to manage Kafka and Elasticsearch
  - Deployment pattern that attempts to automate repeatable tasks that would usually be handled by system operators
  - Make updates to new versions easy

Alarm System Deployment



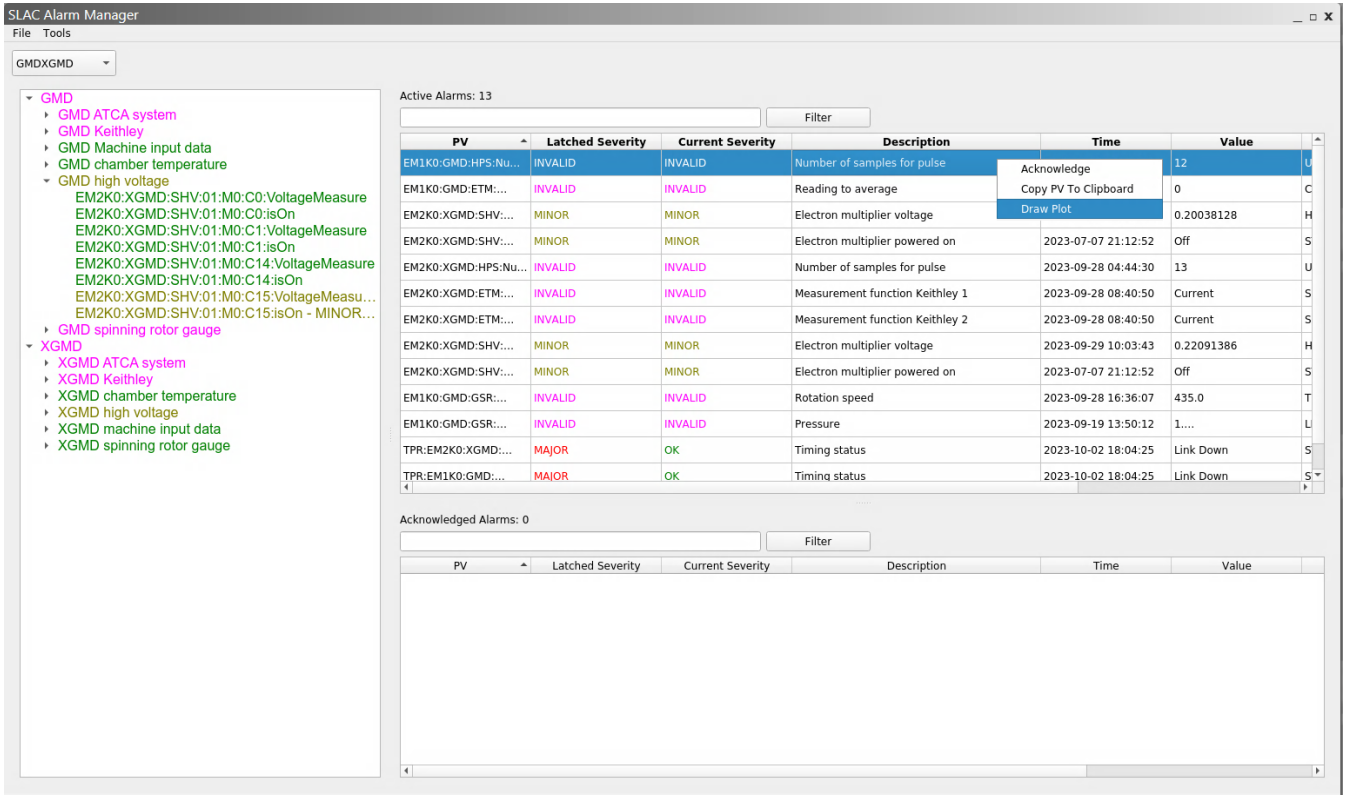
Deployment Updates

Make Editing Alarm Hierarchies as Simple as Possible

- Continuous Integration with GitHub Actions
  - User edits a csv file to reflect their changes and submits a pull request
  - GitHub action validates the change, updates the associated xml file, and lints that file
  - PR still requires approval before being merged

- Continuous Deployment with Kubernetes Sidecar
  - A sidecar is a deployment pattern in which the sidecar container enhances the functionality of the main container by running alongside it in the same pod
  - Our sidecar uses gitsync to monitor the git repository with the xml files
  - When a change is detected, it pulls in the new xml file and uses kcat to issue the alarm server restart command to kafka

Python Alarm Manager



Slack Integration

Conclusion

- Kubernetes learning curve is fairly steep
  - Having dedicated personnel resources helps!
- End user response has been largely positive
- Future projects at SLAC are already slated to use Kubernetes based deployments