ATC Apps Journey to Private Cloud

Matt Skipton World Wide Technology
Advanced Technology Center

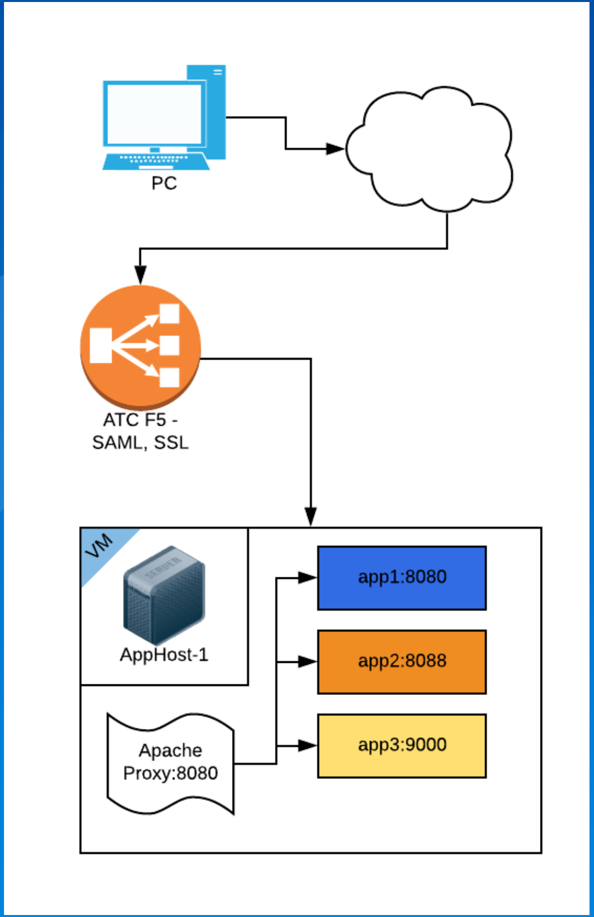
Background

- **The ATC has specific, custom, apps that need to run in the ATC**
  - POC management
  - ATC utilities
  - Platform proxies
  - Lab management
- **Different languages and stacks**
- **Microservices framework creates lots of small apps – eases development and reusability but makes traditional hosting problems worse**

World Wide Technology

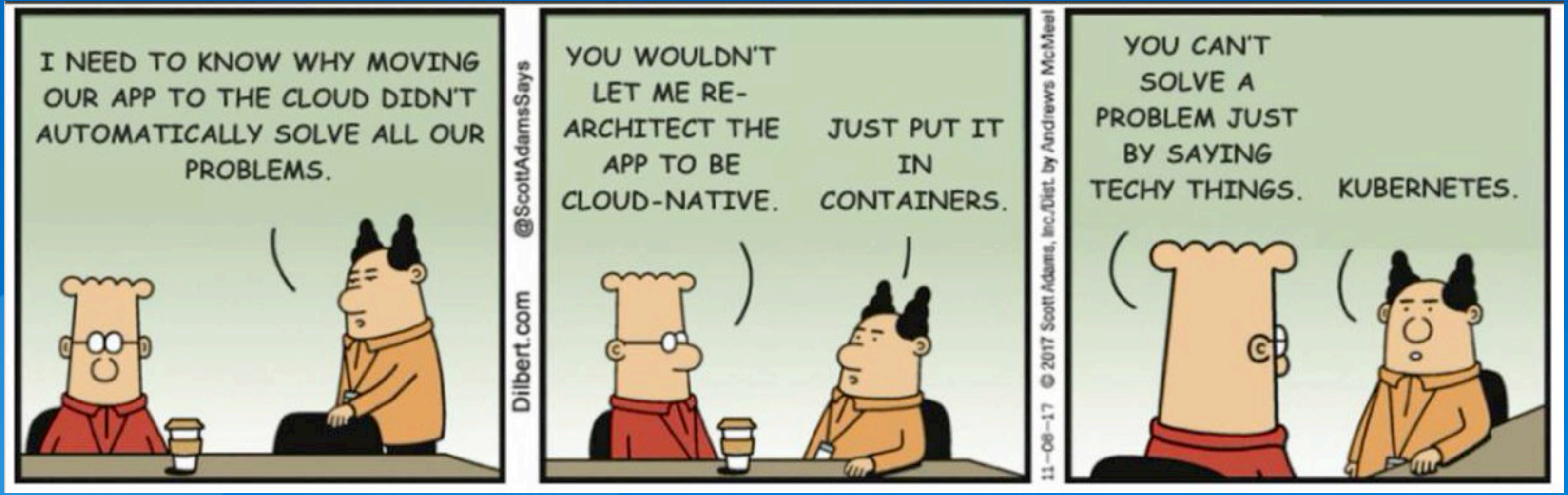
In the beginning….

- **Traditional app hosting**
- **VM's for dev/test/prod**
- **Problems with this method**
  - No app HA/scalibility
  - Different versions of frameworks could conflict
  - Logging & monitoring inconsistent
  - Host(vm) maintenance required all apps to go down
  - VM's with full server OS
  - Mapping ports to apps
  - Complex/Disjointed CD pipeline
  - Complex stack more brittle
  - Slow to provision
  - Microservice routing complexity



World Wide Technology

What should we do?



I NEED TO KNOW WHY MOVING OUR APP TO THE CLOUD DIDN'T AUTOMATICALLY SOLVE ALL OUR PROBLEMS.
YOU WOULDN'T LET ME RE-ARCHITECT THE APP TO BE CLOUD-NATIVE.
JUST PUT IT IN CONTAINERS.
YOU CAN'T SOLVE A PROBLEM JUST BY SAYING TECHY THINGS.
KUBERNETES.
Dilbert.com @ScottAdamsSays
11-08-17 © 2017 Scott Adams, Inc./Dist. by Andrews McMeel

World Wide Technology

Lets Go to Containers!

- **Problems containers solve**
  - Framework conflicts
  - Provisioning speed
  - Simplified CD Pipeline
  - App build config lives with code
  - Consistent deployment on dev laptop and prod
- **Unsolved Problems**
  - No app HA/scalibility
  - Logging & monitoring inconsistent
  - Host(vm) maintenance required all apps to go down
  - Mapping ports to apps

Lets add a container orchestration layer!

- **There are many container orchestration solutions**
- **MesoSphere**
- **Cloud Foundry**
- **Docker Swarm**
- **Kubernetes (the industry leader)**
  - Problems Kubernetes solves
    - App HA / Scalability
    - Host(vm) maintenance without downtime
    - Automatic app communication routing
  - Unsolved problems
    - Logging becomes consistent but still needs to be sent somewhere for aggregation
    - No tracing of packets for troubleshooting
    - Possible security concerns
    - No host monitoring
    - Complex to deploy
    - RBAC is hard!

kubernetes

World Wide Technology

What is Kubernetes (k8s)?

- Started at Google
- Open Source
- Maintained by the supported by the Cloud Native Computing Foundation https://www.cncf.io
- Applications are defined through a declarative model in YAML
- kubectl – CLI management app
- Many GUI management options

- https://www.cncf.io/the-childrens-illustrated-guide-to-kubernetes/

Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of containerized applications.

It groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon 15 years of experience of running production workloads at Google, combined with best-of-breed ideas and practices from the community.

- https://kubernetes.io

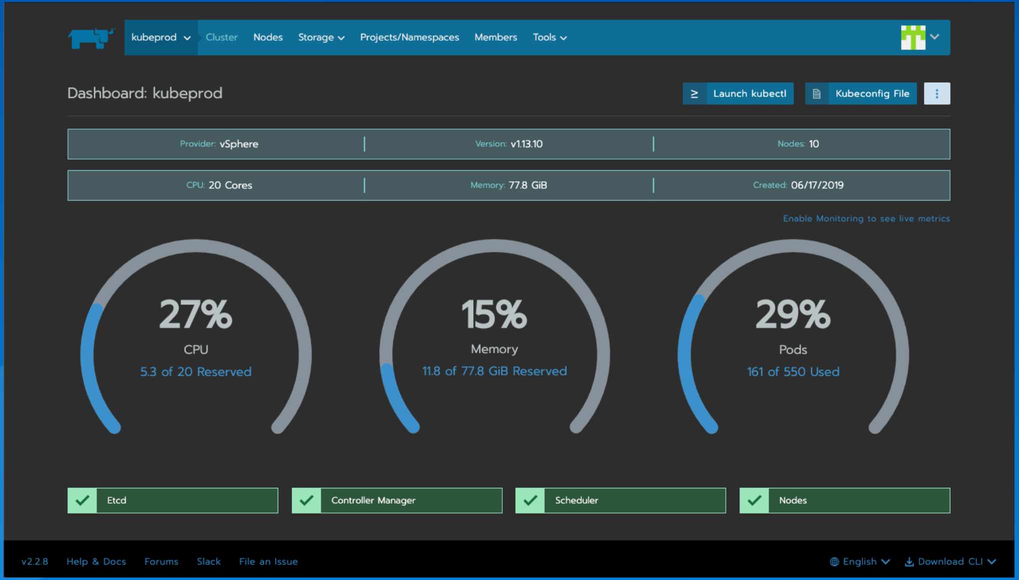
World Wide Technology

Kubernetes it is, but how to deploy & maintain?

- Plain OSS “manual install” Kubernetes
- RedHat OpenShift
- Pivotal PKS
- Amazon EKS
- Azure AKS
- Google GKE
- Ubuntu/Cononical Juju/Conjure-Up
- Terraform
- Rancher

Why Rancher

- **Works with on-prem vSphere and major cloud providers**
- **Easy to deploy / manage many Kubernetes clusters (we have 6 today)**
- **All VM's can be replaced at any time (cattle not pets)**
- **OSS Product is solid, can get support if we want**
- **Provides good management interface for users**
- **Makes Kubernetes RBAC much easier**
- **Provides host monitoring**



Dashboard: kubeprod

Provider: vSphere | Version: v1.13.10 | Nodes: 10

CPU: 20 Cores | Memory: 77.8 GiB | Created: 06/17/2019

Enable Monitoring to see live metrics

27% CPU 5.3 of 20 Reserved

15% Memory 11.8 of 77.8 GiB Reserved

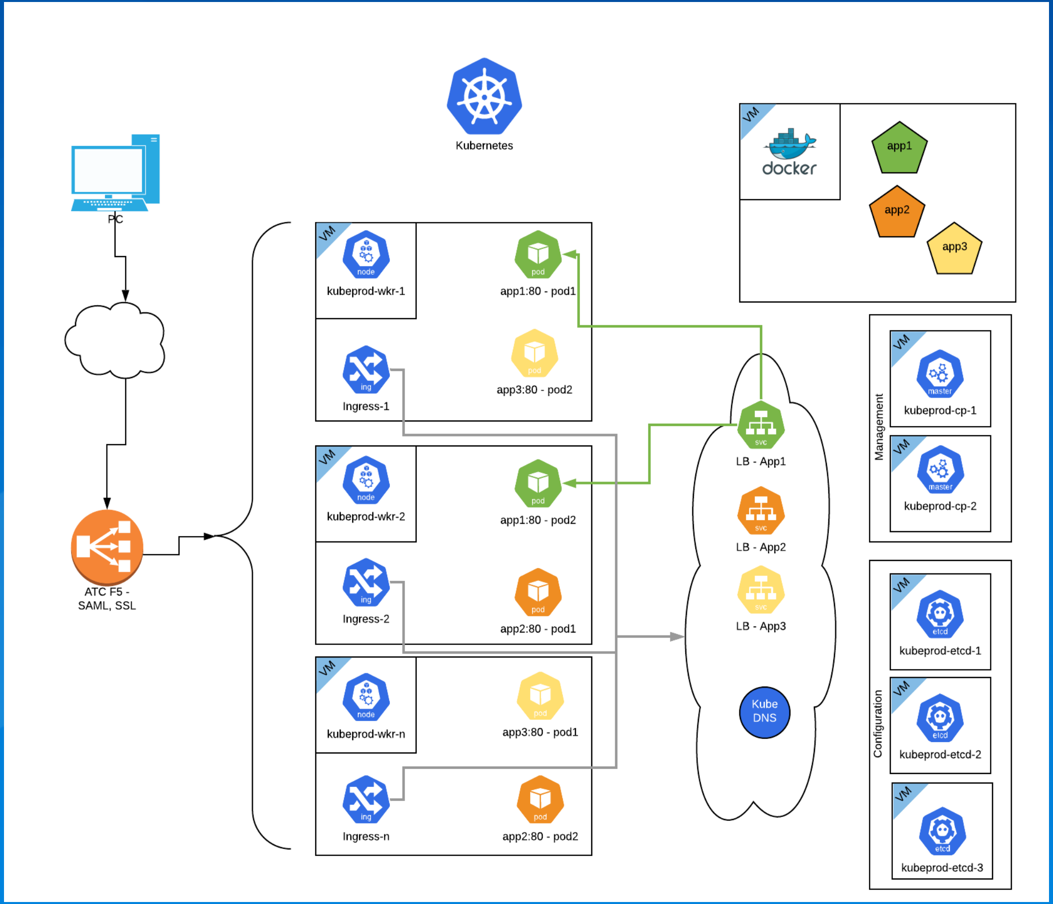
29% Pods 161 of 550 Used

✓ Etcd ✓ Controller Manager ✓ Scheduler ✓ Nodes

World Wide Technology

How we are built

- **Kube nodes in VM's for flexibility**
- **Worker nodes in dedicated subnet to ease load balancing**
- **All apps deploy with multiple replicas (with a couple exceptions)**
- **F5 for initial SSL termination, SAML Auth, and Load Balancing**
- **Custom CI/CD pipeline using Jenkins and custom code**



World Wide Technology

Let’s build a cluster

1. Initial cluster build
2. Deploy a simple web app
3. Deploy from Rancher catalog / helm

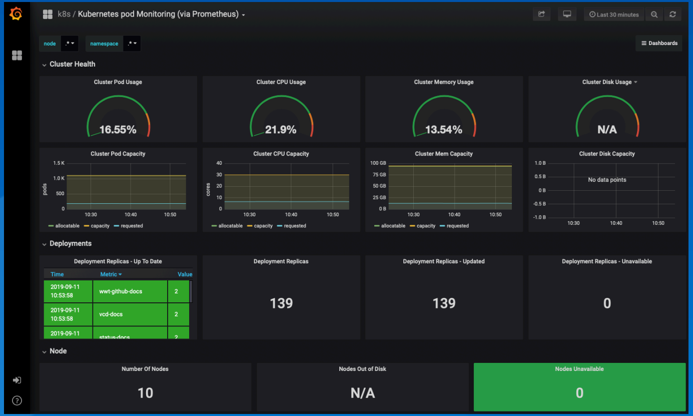
RancherOS

A Lightweight Container Operating System

RancherOS includes the bare minimum amount of software needed to run Docker. Everything else can be pulled dynamically through Docker. RancherOS makes it simple to run containers at scale in development, test, and production. By containerizing system services and leveraging Docker for management, the operating system provides a very reliable and easy to manage container-ready environments.
- https://rancher.com/rancher-os/

Unsolved Problem - Logging stack

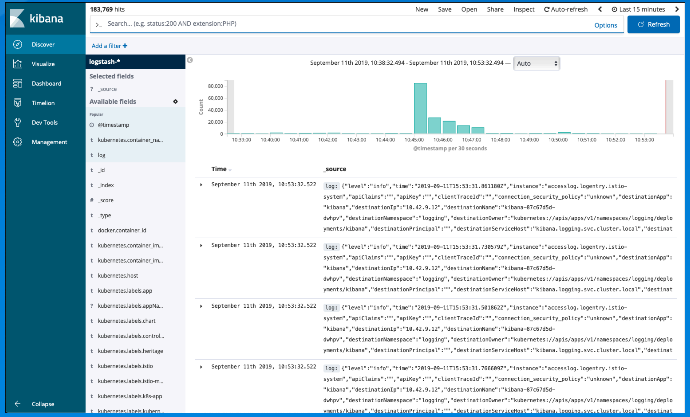
- **We decided on Prometheus and Grafana**
  - standard in the Kubernetes world
  - Open Source and well documented
  - more in-depth monitoring of workloads



Deployment Replicas - Up To Date

Time	Metric	Value
2019-09-11 10:53:58	wwt-github-docs	2
2019-09-11 10:53:58	vcd-docs	2
2019-09-11	status-docs	2

- **We decided on fluentd, Elasticsearch, and Kibana**
  - standard in the Kubernetes world
  - Open Source and well documented
  - Aggregates logs in 1 console for cross referencing



183,769 hits

September 11th 2019, 10:38:32.494 - September 11th 2019, 10:53:32.494

```
Time                                  _source
September 11th 2019, 10:53:32.522    log: {"level":"info","time":"2019-09-11T15:53:31.861180Z","instance":"accesslog.logentry.istio-
                                      system","apiClaims":"","apiKey":"","clientTraceId":"","connection_security_policy":"unknown","destinationApp":
                                      "kibana","destinationIp":"10.42.9.12","destinationName":"kibana-87c67d5d-
                                      dwhpv","destinationNamespace":"logging","destinationOwner":"kubernetes://apis/apps/v1/namespaces/logging/deplo
                                      yments/kibana","destinationPrincipal":"","destinationServiceHost":"kibana.logging.svc.cluster.local","destinat
September 11th 2019, 10:53:32.522    log: {"level":"info","time":"2019-09-11T15:53:31.730579Z","instance":"accesslog.logentry.istio-
                                      system","apiClaims":"","apiKey":"","clientTraceId":"","connection_security_policy":"unknown","destinationApp":
                                      "kibana","destinationIp":"10.42.9.12","destinationName":"kibana-87c67d5d-
                                      dwhpv","destinationNamespace":"logging","destinationOwner":"kubernetes://apis/apps/v1/namespaces/logging/deplo
                                      yments/kibana","destinationPrincipal":"","destinationServiceHost":"kibana.logging.svc.cluster.local","destinat
September 11th 2019, 10:53:32.522    log: {"level":"info","time":"2019-09-11T15:53:31.501862Z","instance":"accesslog.logentry.istio-
                                      system","apiClaims":"","apiKey":"","clientTraceId":"","connection_security_policy":"unknown","destinationApp":
                                      "kibana","destinationIp":"10.42.9.12","destinationName":"kibana-87c67d5d-
                                      dwhpv","destinationNamespace":"logging","destinationOwner":"kubernetes://apis/apps/v1/namespaces/logging/deplo
                                      yments/kibana","destinationPrincipal":"","destinationServiceHost":"kibana.logging.svc.cluster.local","destinat
September 11th 2019, 10:53:32.522    log: {"level":"info","time":"2019-09-11T15:53:31.766609Z","instance":"accesslog.logentry.istio-
                                      system","apiClaims":"","apiKey":"","clientTraceId":"","connection_security_policy":"unknown","destinationApp":
                                      "kibana","destinationIp":"10.42.9.12","destinationName":"kibana-87c67d5d-
                                      dwhpv","destinationNamespace":"logging","destinationOwner":"kubernetes://apis/apps/v1/namespaces/logging/deplo
                                      yments/kibana","destinationPrincipal":"","destinationServiceHost":"kibana.logging.svc.cluster.local","destinat
```

Prometheus Monitoring

Grafana

fluentd

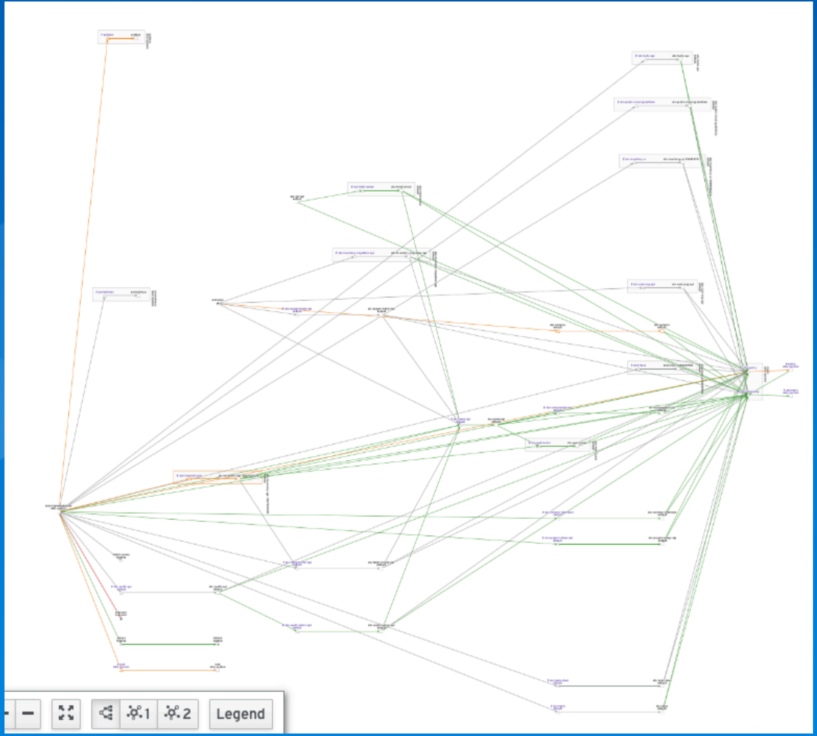
elasticsearch

kibana

World Wide Technology

Unsolved Problem – Understanding Microservices

- **Service Mesh – Istio**
  - Why we use it today
    - Understanding service dependencies
    - Network level tracing (up to layer 7)
    - Impose rules at layer 7 for specific apps
    - http status monitoring (are we erroring often)
    - Additional security
  - Other features for the future
    - Zero Trust environment support
    - Added code level features - circuit breakers



World Wide Technology

Benefits we have seen with this configuration

- **We have re-deployed all nodes many times without service interruption**
- **Clean consistent CD pipeline**
- **Fast deployment of new apps in HA config**
- **Made us aware of application level issues we didn't have visibility to before**
- **Troubleshooting is easier**

World Wide Technology

Conclusions

- Containers and Kubernetes and Service mesh's can solve a lot of app deployment issues and they are advancing at a rapid pace
- They are made up of many different tools that require a lot of choices and automation to work well
- Kubernetes is very complex to deploy but, many of our partners have developed solutions to help get it running
- Service Mesh can add a lot of visibility/functionality/security to your current apps but requires a lot of configuration to implement

World Wide Technology

Definitions

- **Container**
  - A standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. Container images are lightweight, standalone, executable packages. When a container is launched it is isolated as much as possible form the OS which provides security and consistent behavior.
- **Kubernetes**
  - A container orchestration layer. It provides features on top of existing container benefits. Such as: network mapping, load balancing / HA, scalability, storage mapping, and management.
- **Pod**
  - A grouping of containers that execute on the same host. They are managed as 1 unit.
- **Sidecar**
  - A container in a POD that is not the main application. It provides additional services to the main application. Ex: Istio proxy where all network traffic routes through it on the way to the main container.
- **Service Mesh**
  - A grouping of proxies that live next to each application, have configurable rules, low-latency infrastructure, and high throughput. They provide rule driven routing, service discovery, load balancing, encryption, observability, traceability, authentication and authorization.

World Wide Technology

Questions?