Migrating to Cloud Native Applications: City of Ottawa Case Study

Susan Pay

Chris Carty

Tim Pinet

Ottawa

What is Cloud Native

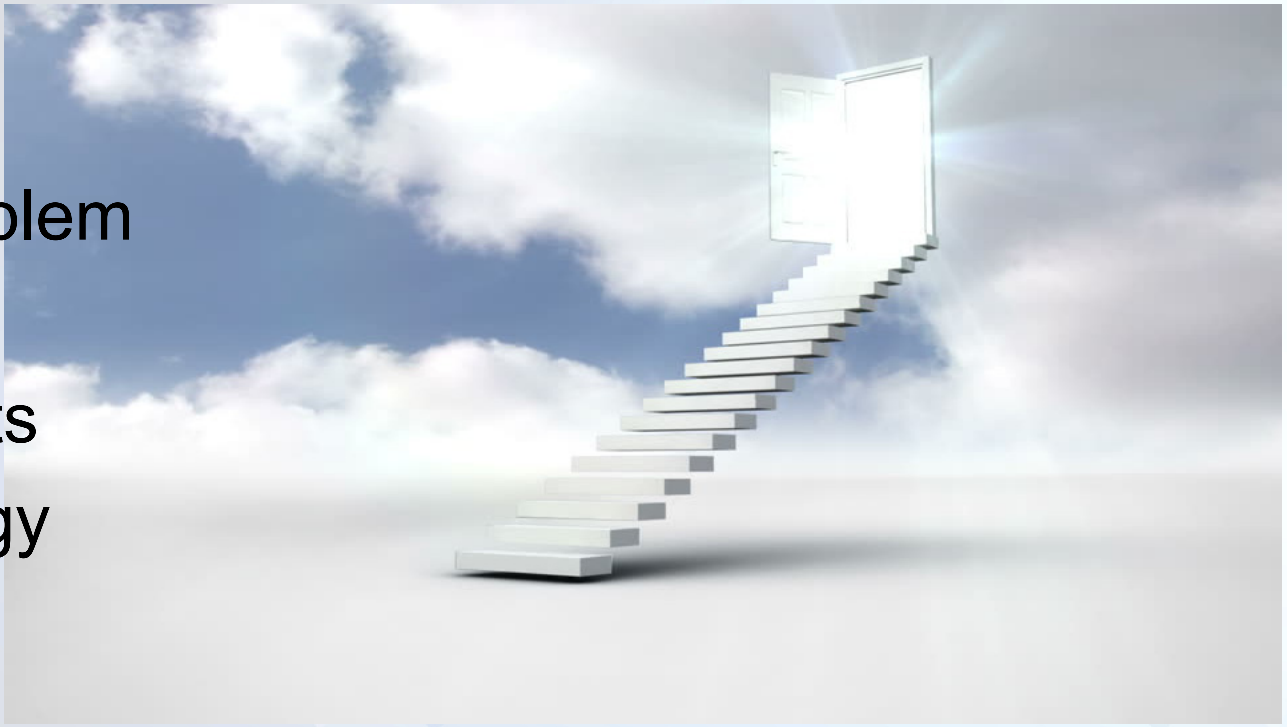
Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.

- CNCF Cloud Native Definition v1.0

TP2

The Journey

- Legacy – the problem
- The last 3 years
  - Recent projects
  - New technology
- Next steps



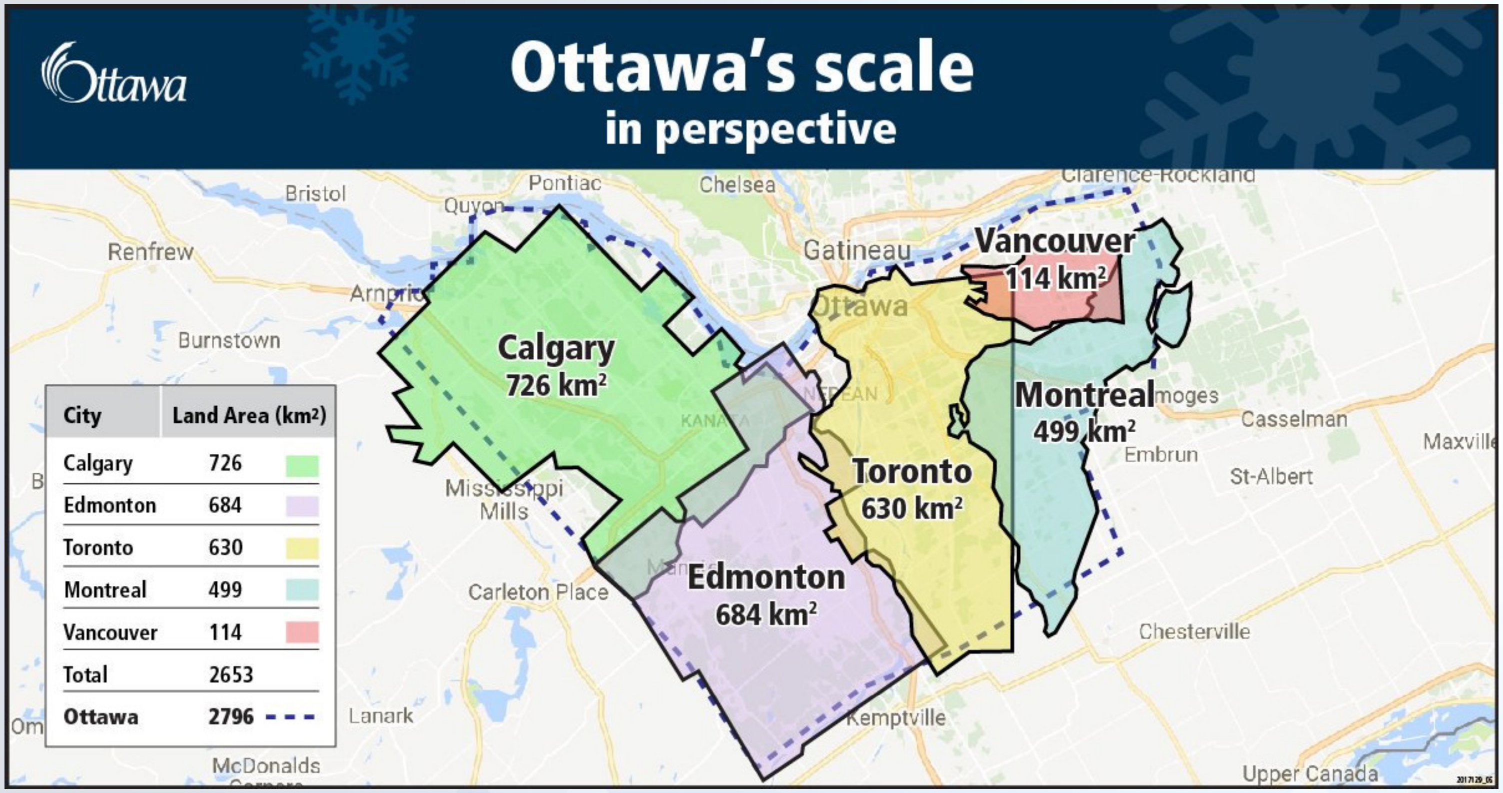
Ottawa

Slide 3

TP2 I put approximate times in the speaking notes but we will need to practice this numerous times to see where we spend too much time vs not enough in others.
Timothy Pinet, 4/28/2019

Our City

- Ottawa is a large city
- Over 1M citizens and also supports Gatineau, at an estimated combined metropolitan size of 1.4M
- Over 100 business lines to provide unique services
- 400 City facilities spread through the region



City	Land Area (km²)	
Calgary	726	🟩
Edmonton	684	🟪
Toronto	630	🟨
Montreal	499	🟦
Vancouver	114	🟥
Total	2653	
Ottawa	**2796**	- - -

City of Ottawa IT

- 17000 staff work for the City, just over 300 of them are IT
- City IT branches: Applications (that’s us!), SAP, Technical Architecture, Infrastructure, Security, Front Line, Project Intake, and special projects office (Office 365).
- Traditionally followed the 70% operations/maintenance, 30% projects model.
- Multiple data centers spread across the city with varying ages of infrastructure.
- Highly demanding client business groups and a technologically advanced citizens asking for new services with quick turnarounds.

Legacy – the problem

- Large application portfolio (400+ applications)
- Keeping our environments sane, similar for consistent testing
- Technical debt
- Many process are still done manually
- Request for resources took weeks and had to go through infrastructure teams
- Aging infra (servers, databases, integration platforms / ESB)

Cloud benefits / risks

Benefits

- Reduced costs
- Flexibility/Scalability
- Options through services
- Improved integrations
- Backup regions, disaster recovery, higher availability
- Costing (Opex vs Capex)
- Security (ISO 27000, ITSG-33, SOC2)
- Modern hardware, continuous improvement
- No need for physical footprint for your own datacenter

Risks

- Increased costs
- Data sovereignty/residency
- Storage can be higher latency/less bandwidth
- Internal staff skillset, misconfiguration
- Downtime (ex: AWS 2017, Azure Jan 2019, GCP Jul 2018, Azure DNS May 2019)
- Security, data leaks, intrusion, zombie mining
- Culture/change
- Vendor lock-in
- Fewer customizations for solutions

Why is the City of Ottawa moving to Cloud

- Ease of deployment and request for servers through code
- Availability of services not available on-prem (compute, storage, AI)
- Container platforms, scalability, self healing
- Overall time to market for new applications (DevOps)
- Reduce reliance on on-prem datacenters
- Public API management to expose datasets to developers and partners
- Potential cost savings by retiring heavily customized apps moving to SaaS offerings
- Some of our application vendors are only offering hosted solutions and not supporting on-prem installs anymore

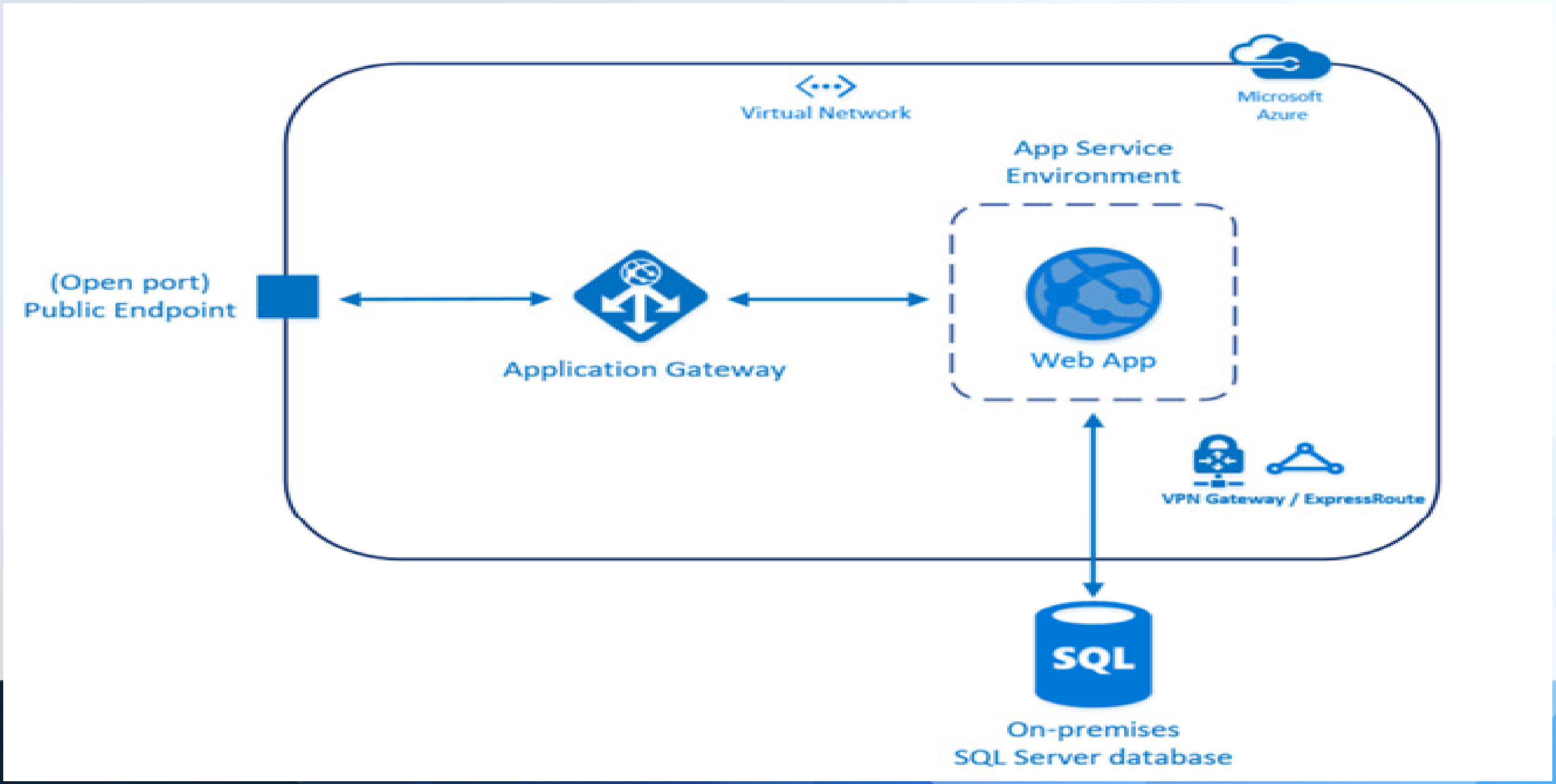
The Last 3 Years

- City’s first mobile App for Open 311 services
- AWS pilot
- ChatBot
- Azure API Management (WIP)
- OpenData (ESRI)
- RedHat Innovation Lab
  - Build an application from cradle grave with an embedded DevOps team in and Agile project towards container based automated deployment and testing.

Recent Projects

- City's first mobile App (2018-2019)
  - Gateway to connect to backend services in City of Ottawa and Mobile application
  - Backend gateway built upon or connect to:
    - Azure App Service and using Web Jobs
    - WAF
    - OneSignal
    - Apigee (open 311 services)

Azure Application Services


Virtual Network
Microsoft Azure
App Service Environment
(Open port) Public Endpoint
Application Gateway
Web App
VPN Gateway / ExpressRoute
SQL
On-premises SQL Server database

The Good….

- PaaS
  - Easy to configure: infrastructure as code
  - Easy to use already built services
  - Performance: Easy to scale up and down, easy to scale out
  - Security: Using Azure Key Vault
  - No more OS patching
- Azure DevOps
  - Easy to deploy
    - ARM template
  - CI/CD Pipeline
- Good debug and monitoring tools available: Application insights
- Keep all environments in sync
- SLA and Microsoft Support

Lessons Learned

- Skill set need to be adjusted, more focus on integrations and learn how to use and configure Azure Services
  - WAF
  - New kids on the block, sometimes requirements are not clear
- Less control (Owners Vs Renters)
  - Trouble shooting
  - Upgrade platform impacts
- To Be Continued…….

AWS Pilot Project (2017-2018)

- AWS pilot
  - Cost
  - Move a few public facing applications which need to be upgraded/replaced
  - Not complex applications
  - Develop new applications using the services in AWS
  - Web applications
  - EC2

AWS Pilot Project Findings 2017-2018

Step 1. Choose applications to move to the cloud

- Make sure the application data meet the requirements (i.e. data is sanitized and public data)
- Pick a small application and try it out (Lobbyist Registry)
- Pick applications more cloud ready to move to the cloud (GIS server)
- Develop new applications using the services in AWS (such as SNS) (was on hold due to lack of strong business cases)

Step 2: Understand the architecture of the applications and information gathering

database (what kind? How big? traffic patterns ?)

applications (web server? Dependencies? Languages? Storages?)

Physical Server(RAM, CPU, storage? Uptime?)

Step 3: Collect Security Requirements

AWS Pilot Project Findings 2017-2018

Step 4: Determine Methods used moving application to the cloud:

- Lift and shift (used for GIS server)
  - single EC2
  - break out 1 service, e.g. RDS (database)
  - auto scaling, auto backup
- Put application in the container first (i.e. so it is easier for DevOps later on). This could mean modernization of the application first then deploy it to the cloud.

Step 5: Setup development environments

Step 6: Test

Findings

1. First application has already an cloud friendly version available.
   lift and shift worked
   cloud instance as backup and recovery
2. Second Application
   Database is Oracle, we have to change the database due to the cost
3. If the application is too old, we have to modernize it first
4. AWS Chatbot experience (AWS Lambda)

The Good…

1. Easy to configure EC2 Service, some free tier
2. Easy to upgrade and patching (AMI with application, EC2,)
3. Performance (auto scaling and Elastic Beanstalk)
4. Monitoring (using monitoring services)
5. Backup and Recovery
6. Security

Lessons Learned

1. Complex Security (IAM, different layers and it is shared with AWS)

    Use CloudTrail to security auditing.

    – Better cloud security training before dive too deep.
    – Use MFA

2. Alone

3. Setup Governance earlier in the project

4. Use Trust advisor to make sure use AWS best practices. (need a support plan)

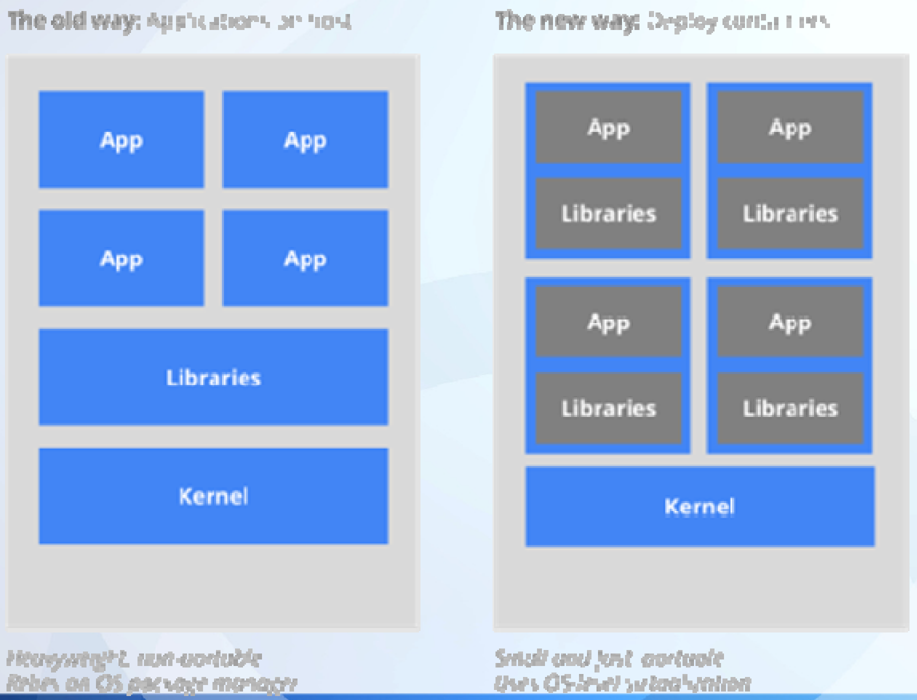
New technology

KUBERNETES AND CONTAINERS

Why are we adopting it?

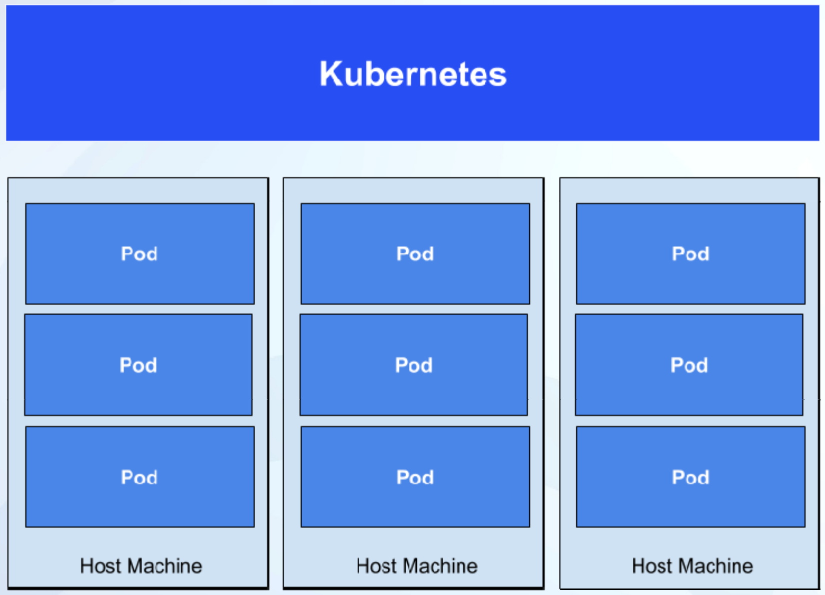
What Are Containers

- Virtualized OS
- Shares Kernel with host
- Lightweight
- Run only what you need
- Dependency Isolation
- Think of them as super charged process. One process per container


The old way: Applications on host
App
App
App
App
Libraries
Kernel
Heavyweight, non-portable
Relies on OS package manager
The new way: Deploy containers
App
App
Libraries
Libraries
App
App
Libraries
Libraries
Kernel
Small and fast, portable
Uses OS-level virtualization

What is Kubernetes

- Production Grade Container Orchestrator
- Supported on all major cloud platforms such as Azure, AWS, Google
- Developed and Open sourced by Google
- Self-Healing
- Auto-Scaling
- Operational Metrics
- Automated Rollouts and Rollbacks



Why Kubernetes?

- Common platform
  - On-Prem
  - Cloud (AWS, Google, Azure)
- Portability between environments
- Enterprise grade container orchestrator
- Central to our container adoption strategy
- Optimize resource usage

Ottawa

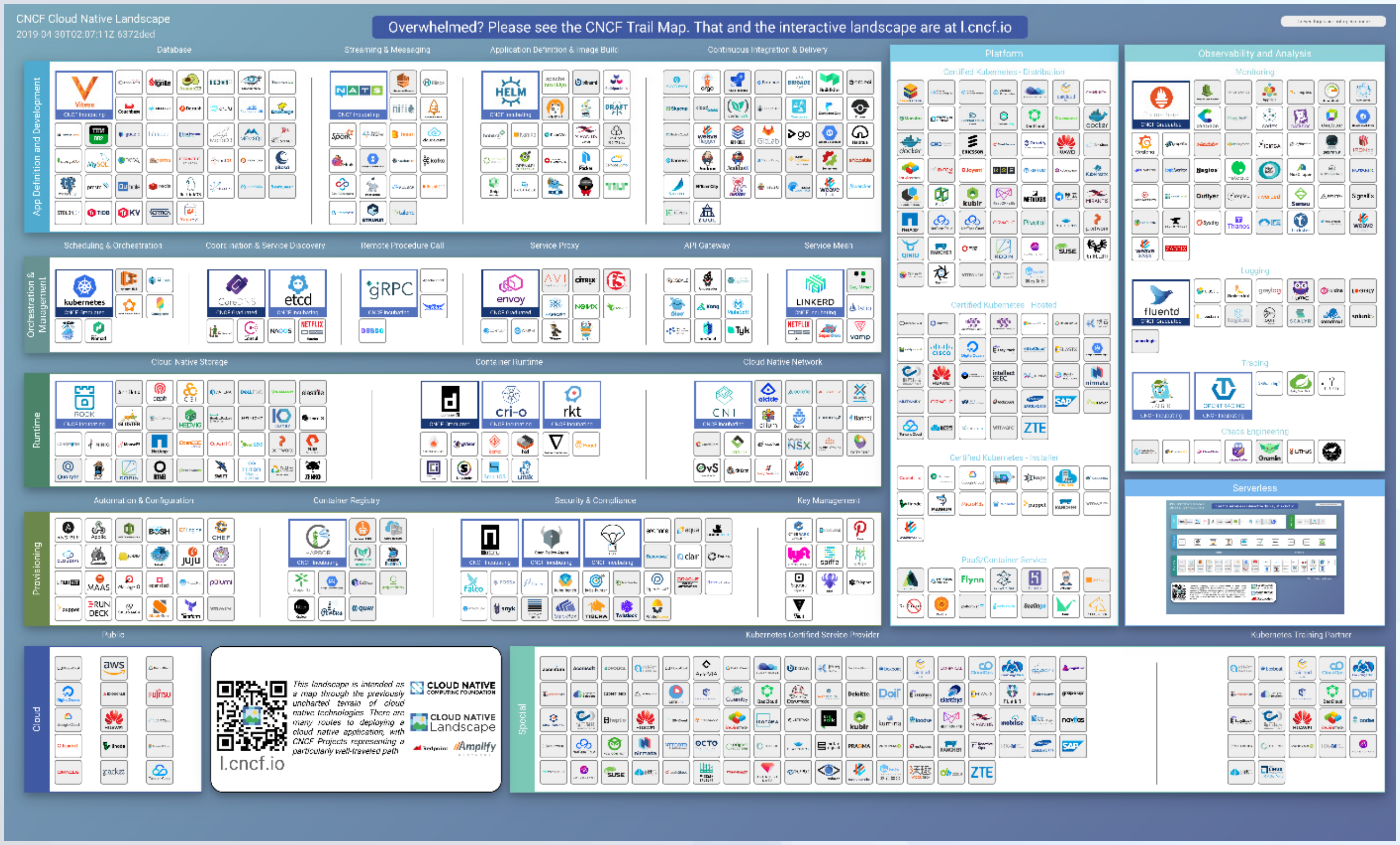
CHALLENGES

Lessons we've learned adopting k8s

Spoiler Alert: It's hard!

CLOUD NATIVE
COMPUTING FOUNDATION

- Serves as the vendor-neutral home for many integral Cloud Native OSS Projects
  - Kubernetes, Containerd, CRI-O, Harbor, Prometheus, Open Policy Agent, Envoy, gRPC
- One of the Linux Foundation’s largest sub-foundations.
- Governing Board: Microsoft, AWS, Red Hat, Huawei, VMware, Rancher, Gitlab, Google, JPMorgan Chase, SAP, Oracle


CNCF Cloud Native Landscape
2019-04-30T02:07:11Z 6372ded
Overwhelmed? Please see the CNCF Trail Map. That and the interactive landscape are at l.cncf.io
App Definition and Development
Database
Streaming & Messaging
Application Definition & Image Build
Continuous Integration & Delivery
Orchestration & Management
Scheduling & Orchestration
Coordination & Service Discovery
Remote Procedure Call
Service Proxy
API Gateway
Service Mesh
Runtime
Cloud Native Storage
Container Runtime
Cloud Native Network
Provisioning
Automation & Configuration
Container Registry
Security & Compliance
Key Management
Platform
Certified Kubernetes - Distribution
Certified Kubernetes - Hosted
Certified Kubernetes - Installer
PaaS/Container Service
Observability and Analysis
Monitoring
Logging
Tracing
Chaos Engineering
Serverless
Cloud
Public
Special
Kubernetes Certified Service Provider
Kubernetes Training Partner
This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path.
l.cncf.io
CLOUD NATIVE COMPUTING FOUNDATION
CLOUD NATIVE Landscape
Amplify

Ottawa

CNCF Cloud Native Landscape

Our Requirements

- Vendor Neutral as possible in order to reduce lock-in
- Enterprise Support
- Run On-Premises or in the Cloud
- LDAP Integration
- Support for both Linux Containers and Windows
- Supports RBAC (Role Based Access Control), and Pod Security Policies

Other technical considerations

- Brand new technology
  - Design patterns
  - Workflow
  - Architecture
- Build the infrastructure
- Test the infrastructure
- Training staff
- Developer Workflow?
- What tools do we choose?
- How do we manage/support it

- Cluster design
  - Many Small Clusters
  - Fewer Larger Clusters
  - Cluster per environment (DEV/QA/Prod)
  - Namespaces per environment
- Where will you be deploying?
  - Roll your own artisanal Kubernetes
  - Use a managed solution
  - Both?
- New Security Concerns

WHERE ARE WE

Our Solution So Far....

Tools So Far

- Rancher for Kubernetes Management
  - Can handle on-prem or cloud deployments (Azure, AWS, Google)
  - Vanilla Kubernetes
  - Provides a single pane of control for all clusters and AD integration
- Prometheus For Monitoring
- Currently Elasticsearch, Fluentd and Kibana stack for logging (but may change)
- Project Harbor for on-prem container registry
- Pod Security Policies and Network Policies
- Making use of existing infrastructure and technology
  - Vmware
  - NetApp
  - AD

The Good

- Strong configuration management
  - Easily migrate applications between environments
- Security standards are more easily applied
  - Open Policy Agent
  - Container Security Scanning
- Reduced reliance on “Servers”
  - Containers are much more flexible and portable
  - Reduces “it works on my machine” scenarios
- Quickly prototype apps
  - Local clusters are easy to spin up
- Increased observability into applications

Lessons Learned so far

- Increment and keep it simple to start
  - There are lots of tools out there. No is temporary, yes is permanent.
- Narrowing down common deployment tools is hard.
  - Helm, Kustomize, Boiler plate config templates
- Keep Cluster config in source control. Comes in very handy for disaster recovery.
- Keep application configuration files separate from Source Code.
- Many Small Clusters. Helps with separate of duties, and avoids major outages.
  - Different security policies based on Cluster focus.
  - Ie CICD Cluster, Cluster Per Env, Sensitive workload

Where do we go from here?

- Take advantage of Cloud resources (Azure)
- Improved Security and Policy Management
  - KeyManagement (CyberArk Conjur)
  - Policy Management/Enforcement (Open Policy Agent)
- Common Development Pipeline/ToolSet
  - CICD (Jenkins, Azure DevOps, Gitlab, Weaveflux)
  - Helm, Kustomize
- Training, On-Board more staff

Collaborate!

- Do not do this in a silo
- A successful deployment will need support from Security, Infrastructure, and Developers

Great team progress

- 2017
  - team of 4 application developers for technology pilots
- 2019
  - 75% of the City's IT staff are working on cloud in some way shape or form. Applications, Infrastructure, and Security working together.
  - Staff are energized learning new technologies.
  - Faster TTM so we can get applications delivered faster to our business clients and citizens.
  - Reducing our operations/maintenance ratio and focusing more on value add projects.
  - Standardizing on toolsets and processes.

Next Steps

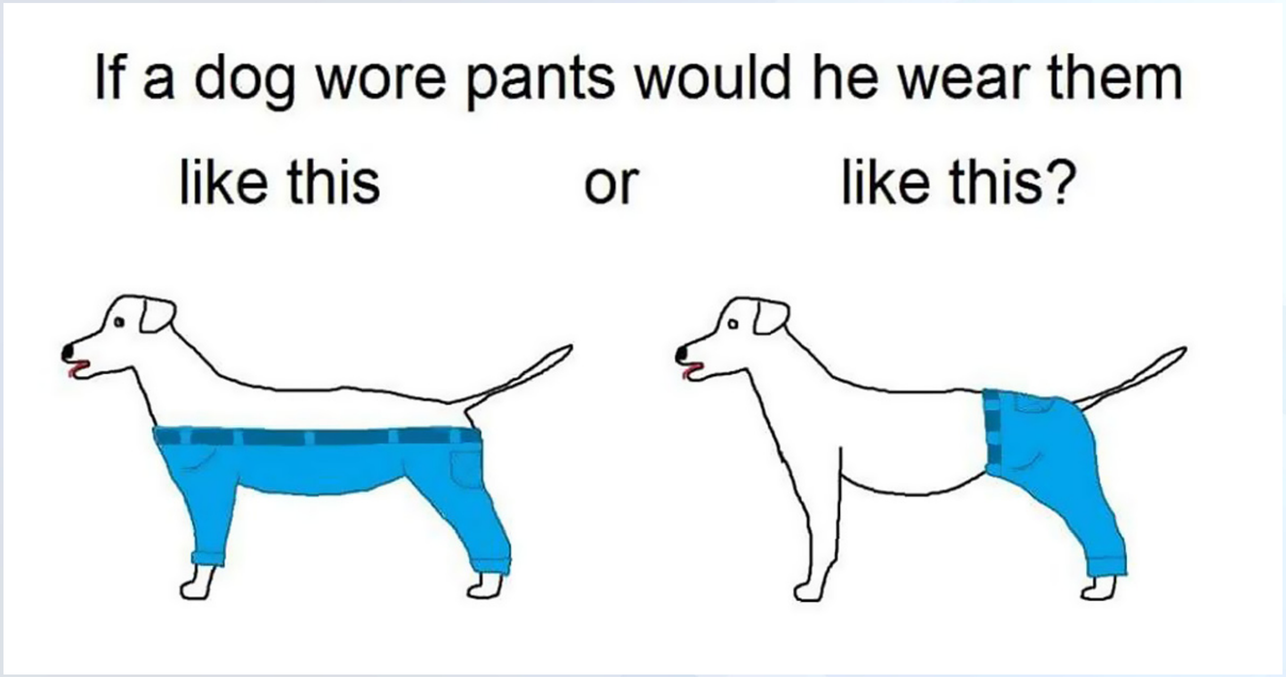
- We are in the beginning of the journey but happy with our progress!
- Applications team Next Steps:
  - DevOps (new dedicated team formation – 2019)
  - Cloud Architects, Cloud Data Architects, Azure engineers (new jobs)
  - Develop new applications using new platforms/services and deploy to public cloud
  - Continue to migrate existing applications to use cloud services
  - Azure API management project (2019)
- Other teams in City IT:
  - Office 365
  - Azure infrastructure for servers
  - Colocation data center for Hybrid
  - SaaS for major enterprise software

Ottawa

The Journey -- Recap

- Legacy – the problem (big City, lots of sprawl)
- The last 3 years
  - Recent projects (Mobile app, Azure API, AWS pilots)
  - New technology (Azure, Kubernetes/containers)
- Next steps (new teams, more production workloads)

Questions?



Contact

- Susan Pay (susan.pay@ottawa.ca)
- Chris Carty (christopher.carty@ottawa.ca)
- Tim Pinet (tim@pinet.ca)