Cloud Foundations 1 & 2

Canturk Isci
IBM Research, NY
@canturkisci

Bilkent University
Tue Aug 16, 10:30 AM
Thu Aug 18, 10:30 AM
Operational Visibility and Analytics
Designed for Cloud

Canturk Isci
IBM Research, NY
@canturkisci

Cloud Foundations 1
Bilkent Univ
Tue Aug 16, 10:30 AM
ABSTRACT: We are seeing an accelerating growth in cloud platforms, runtimes, and programming models. Cloud discussion has shifted from utility and density to cloud-native services and design patterns. Emerging cloud services allow users to define and provision complex, distributed systems with unprecedented simplicity and agility. With the push of a button entire stacks of software can be instantiated within minutes with various configurations and customizations. Automation, continuous integration and delivery further simplify the entire lifecycle management of modern born-on-the-cloud applications. These advances also bring in new research challenges. Operational visibility into the complex, distributed applications, cloud runtimes and the underlying infrastructure is becoming a persistent pain point across end-users and providers especially for security applications. As system and configuration complexity grows, data-driven operational analytics for security, compliance, configuration and resource management emerge as key areas of focus, where traditional solutions remain ineffective or insufficient.

In this talk I will present an overview of the cloud evolution, emerging runtimes and design patterns. I will describe the challenges arising from this evolution and where existing techniques fall short. I will then present our work on cloud operational visibility and analytics that aims to address some of these challenges. I will describe a unique approach to leveraging cloud abstractions and implementation principles to achieve unmatched deep and seamless visibility into cloud instances, and using this deep visibility in developing operational and security analytics for the cloud. I will overview two outcomes of this approach, Agentless System Crawler and the Vulnerability Advisor service. I will discuss our journey developing the foundations of the visibility and security services for IBM Containers. I will share our experiences working with a production cloud and the key real-world use cases. I will provide an overview of our current research directions, open problems and opportunities in this area.

Bio: Dr. Canturk Isci is a Research Manager and Master Inventor in IBM TJ Watson Research Center, Yorktown, US, where he leads the Cloud Monitoring, Operational and Security Analytics team. He currently works on deep introspection based monitoring techniques for cloud, and their application to novel operational, security and DevOps analytics. He is the technical lead for IBM Vulnerability Advisor for Containers and for Agentless System Crawler.

His research interests include operational visibility, analytics and security in cloud, virtualization, energy-efficient and adaptive computing. Prior to IBM Research, Dr. Isci was a Senior Member of Technical Staff at VMware, where he worked on distributed resource and power management. He has 50 academic papers and 30 issued or pending patents. Dr. Isci has a B.S. from Bilkent University, an M.Sc. with Distinction from University of Westminster, UK and a Ph.D. from Princeton University, US.
Introductions

Virtualization RM

Datacenter Energy and Thermal

Server Power States (S3)

Data Center Robots

Systems as Data

Cloud OpVis & Sec Analytics
Introductions++

Cloud Foundations and Programming Models: From Infrastructure to ML/DL and Serverless

- Deep Learning As a Service
  (DLaaS, Watson ML)

- Microservices Framework
  (Istio, Tracing, Analytics)

- Cloud DevOps
  (CI/CD, Pipelines, A/B, Canary, Toolchains)

- Next-gen Cloud Platform
  (Containers, Kubernetes, Netw)

- Next-gen Cloud Infrastructure
  (HW, Netw, DC, Accelarators, …)

- API Ecosystems
  (API Harmony)

- Operational Visibility and Analytics
  (Deep Introspection, Crawlers and VA)

- Serverless Computing
  (OpenWhisk)

- Security
Introductions++

Cloud Foundations and Programming Models: From Infrastructure to ML/DL and Serverless

- Next-gen Cloud Infrastructure
  (HW, Netw, DC, Accelerators,...)
- Cloud DevOps
  (CI/CD, Pipelines, API, Canary, Toolchains)
- Microservices Framework
  (Istio, Tracing, Analytics)
- Deep Learning As a Service
  (DLaaS, Watson ML)
- Operational Visibility and Analytics
  (Deep Introspection, Crawlers and VA)
- Serverless Computing
  (OpenWhisk)
- API Ecosystems
  (API Harmony)
- Security
- Security
Next Time

Cloud Foundations and Programming Models: From Infrastructure to ML/DL and Serverless

Deep Learning as a Service
(MLaaS, Watson ML)

API Ecosystems
(API Harmony)

Security

Operational Visibility and Analytics

Deep Integration, Manageability, and CI/CD

Microservices Framework
(Istio, Tracing, Analytics)

Cloud DevOps
(CI/CD, Pipelines, A/B, Canary, Toolchains)

Serverless Computing
(OpenWhisk)
<table>
<thead>
<tr>
<th>Year</th>
<th>Technology/Innovation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1970</td>
<td>Intel’s Accidental Revolution</td>
<td>4004 for Calculators → x86 GPPs</td>
</tr>
<tr>
<td>2000</td>
<td>Transmeta</td>
<td>X86 compatible Low-power processors → BT for x86</td>
</tr>
<tr>
<td>2002</td>
<td>VMware</td>
<td>Win on NIX with BT → Server virtualization, drives GPP virt (Vt-x, etc.)</td>
</tr>
<tr>
<td>2004</td>
<td>Xen, KVM</td>
<td>Commoditizing virtualization → Paves the way for cloud</td>
</tr>
<tr>
<td>2007</td>
<td>AWS</td>
<td>Bookstore/Web Svcs w SOI obsession → IaaS for the world</td>
</tr>
<tr>
<td>2010</td>
<td>CloudStack, OpenStack, etc.</td>
<td>Commoditizing cloud/IaaS → cloud platforms</td>
</tr>
<tr>
<td>2013</td>
<td>Docker (DotCloud)</td>
<td>Dev focused hosting svc → New way of SW delivery → container cloud</td>
</tr>
<tr>
<td>2015</td>
<td>Kubernetes (Google)</td>
<td>Warehouse computer → Imctfy, Borg, Omega → K8s cloud orchestration</td>
</tr>
<tr>
<td>2017</td>
<td>Serverless (AWS, IBM, MS, G)</td>
<td>FaaS/IFTTT/Pipelines → Serverless, pay-per-use compute</td>
</tr>
</tbody>
</table>
Cloud Evolution: Trends – 1. Virtualization
Cloud Evolution: Trends – 2. IaaS Cloud
Cloud Evolution: Trends – 3. Containers
Cloud Evolution: Trends – 4. Orch., Runtimes & PMs

Jul 2010
- openstack: 6
- kubernetes: 0

Jan 2016
- serverless: 3
- microservices: 33
Cloud Evolution: Trends – Global Summary
Cloud Evolution: Trends – Global Summary
Challenges & Opportunities & Hype Level
Containers

Standardized **packaging** and **shipping** for applications and all dependencies

**Run** across platforms without changes, all inclusive requirements

Collection of processes isolated by kernel via **cgroups** and **namespaces**

Have been around for a while, **lxc** made consumable and **docker** made popular
Containers – a brief history

- **Early history:**
  - Solaris Zones debuts OS level virtualization – 2004
  - IBM Workload Partition (WPAR) for AIX - 2007

- **Linux history:**
  - cgroups project donated by Google, leads to Linux containers (LXC) - 2007
  - Docker tools open sourced - 2013

- **Our history:**
  - Contributions to IBM Containers and Docker 2014
  - Container Service GA on Bluemix, June 2015
  - Operational Visibility in Containers 2015
  - Security Analytics w Vulnerability Advisor 2015
  - Kubernetes Service GA 2017
Container: Namespaces + cgroups + overlay file system + image format
Image: FS contents of container; as a layered FS; images can share layers
Registry: Where images are; Docker Hub, DTR, Private Registry
Engine: Daemon that manages container lifecycle
Orchestration: How cluster of containers are placed and managed across engines
Kubernetes in a Nutshell

```yaml
apiVersion: apps/v1beta1
kind: StatefulSet
metadata:
  name: web
spec:
  serviceName: "nginx"
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: gcr.io/google_containers/nginx-slim
          ports:
            - containerPort: 80
          name: web
          volumeMounts:
            - name: www
              mountPath: /usr/share/nginx/html
```

Diagram:

- **Master**
  - Scheduler
  - Controller-manager
  - API-server

- **Pods**
  - Pod: Nginx-0
  - Pod: Nginx-1

- **Docker** (container-engine)

- **Kubelet** (container-engine)

- **Desired state**
- **Actual state**
Kubernetes Resources

Diagram:

- **Cluster**
  - Belongs to 1
  - Groups: 1..*

- **Node**
  - Scheduled on
  - Hosts: 0..*

- **Replication controller**
  - *
  - 1

- **Pod**
  - Represents
  - Proxied by 0..1
  - 1
  - 2..*

- **Service**
  - 0..*

- **Container**
  - 2..*
Besides Docker & Kubernetes

Docker: 94%
LXC: 15%
rkt: 10%
BSD Jails: 5%
Solaris Zones: 5%
Other: 5%
LXD: 4%
Don’t know: 0%

Scheduling Tools Used for Containers

- Kubernetes: 36%
- Marathon: 31%
- Docker Swarm: 21%
- Chronos: 14%
- Mesosphere DCOS: 12%
- Consul: 9%
- Homegrown/Custom: 6%
- etcd: 5%
- AWS ECS: 4%
- Nomad: 4%
- Rancher: 3%
- Other: 3%
- Ansible: 2%
- Apache Aurora: 2%
- Engine Yard: 2%
- HTCCondor: 2%
- Mesos: 2%
- OpenShift: 2%
- Rundeck: 2%

Source: The New Stack Survey, March 2016. What do you use for scheduling for containers? Please select all that apply. n=121. Choices with less than two responses are not shown.

Figure 14: Open source Kubernetes, Marathon and Swarm are commonly used to schedule containers.
Back to Cloud Evolution & Challenges

What is Great
- Density
- Scale
- Portability
- Repeatability
- Speed

What Needs Work
- Visibility
- Operational Insight
- Cloud-native Security

- Modernization of IT infra and SW delivery
- Complex made simple
- Unprecedented efficiency and TTV
- Lots of shiny toys across IT lifecycle

BUT:
- Visibility into our environments remains an issue
- Also lots of shiny toys for monitoring & analytics
- Still, mostly based on traditional IT Principles!
Our Work: Built-in Op Visibility & Analytics Designed for Cloud

- Provide unmatched **deep, seamless visibility** into cloud instances
- Drive operational insights to solve real-world pain points
- Provide unmatched **deep, seamless visibility** into cloud instances
- Drive operational insights to solve real-world pain points
- Provide unmatched **deep, seamless** and **unified visibility** into **ALL** cloud instances
- Drive operational insights to solve real-world pain points
Seamless: Built-in Operational Visibility for Containers

"Users do not have to do anything to get this visibility. It is already there by default"
Seamless: Built-in Operational Visibility for Containers

"Users do not have to do anything to get this visibility. It is already there by default"
Why Agentless System Crawlers

Key Advantages

- Monitoring built into the platform not in end-user systems
- No complexity to end user (They do nothing, all they see is the service)
- No agents/credentials/access (nothing built into userworld)
- Works out of the box
- Makes data consumable* (lower barrier to data collection and analytics)
- Better Security* for end user (No attack surface, in userworld)
- Better Availability* of monitoring (From birth to death, inspect even defunct guest)
- Guest Agnostic (Build for platform, not each user distro)
- Decoupled* from user context (No overhead/side-effect concerns)
- Monitoring done right for the processes of the Cloud OS
Crawler: How it Works for VMs

- Leverage VM Introspection (VMI) techniques to access VM Mem and Disk state (We built bunch or our own optimizations that make this very efficient and practical)
- Can even remote both (decouple all from VM and host)
- Almost no new dependencies on host
- Currently support 1000+ kernel distros
Crawler: How it Works for Containers

- Leverage Docker APIs for base container information
- Exploit container abstractions (namespace mapping and cgroups) for deeper insight
- Provide deep state info at scale with no visible overheads to end user

1) Get visibility into container world by namespace mapping
2) Crawl the container
   (Crawler dependencies still borrowed from host. No need to inject into container!)
3) Return to original namespace
4) Push data to backend index
Vulnerability Advisor

This Session

- **Vulnerability Advisor, Policy Mgr**
- Go to Bluemix Catalog
- See VA Image Status
  (Safe, Caution, Blocked)
- Go to Create View
- Explore Status Details
  (Vulnerabilities, Policy Violations)
- Browse Policy Manager
  (Policy Settings, Deployment Impact)
- Change Org Policies
- Override Policies
  (Don’t do it)
- See Weak Password Discovery
- Update Image in Local Dev
- Fix Policy Violation

Previously

- **Built-in Monitoring & Logging**
- We just did that one...
Vulnerability Advisor Report

Login to Bluemix London (https://console.eu-gb.bluemix.net/)

Go to Catalog and Look for Containers
Hover over containers to see VA verdict:
Safe to Deploy | Deploy with Caution | Blocked

Click on Image to go to Create View
See Verdict Details and Explore Options

View Vulnerability Advisor Report:
Discovered Vulnerabilities | Policy Violations
Vulnerability Advisor Report

Login to Bluemix London (https://console.eu-gb.bluemix.net/)

Go to Catalog and Look for Containers
Hover over containers to see VA verdict:
Safe to Deploy | Deploy with Caution | Blocked

Click on Image to go to Create View
See Verdict Details and Explore Options

View Vulnerability Advisor Report:
Discovered Vulnerabilities | Policy Violations
Policy Manager and Deployment Impact

Login to Bluemix London (https://console.eu-gb.bluemix.net/)

Go to Catalog and Look for Containers
Hover over containers to see VA verdict:
Safe to Deploy | Deploy with Caution | Blocked

Click on Image to go to Create View
See Verdict Details and Explore Options

View Vulnerability Advisor Report:
Discovered Vulnerabilities | Policy Violations

Policy Manager and Deployment Impact
Policy Manager and Deployment Impact

Login to Bluemix London (https://console.eu-gb.bluemix.net/)

Go to Catalog and Look for Containers
Hover over containers to see VA verdict:
Safe to Deploy | Deploy with Caution | Blocked

Click on Image to go to Create View
See Verdict Details and Explore Options

View Vulnerability Advisor Report:
Discovered Vulnerabilities | Policy Violations

Policy Manager and Deployment Impact
Change Org Policy and Observe Impact
VA / Security Analytics vs. Cloud Challenges:
Why VA / Security Analytics:

### Cloud Challenges 2016 vs. 2015

<table>
<thead>
<tr>
<th>Challenge</th>
<th>2015</th>
<th>2016</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lack of resources/expertise</td>
<td>28%</td>
<td>32%</td>
</tr>
<tr>
<td>Security</td>
<td>28%</td>
<td>29%</td>
</tr>
<tr>
<td>Compliance</td>
<td>25%</td>
<td>26%</td>
</tr>
<tr>
<td>Managing multiple cloud services</td>
<td>25%</td>
<td>26%</td>
</tr>
<tr>
<td>Managing costs</td>
<td>26%</td>
<td>24%</td>
</tr>
<tr>
<td>Complexity of building a private cloud</td>
<td>24%</td>
<td>24%</td>
</tr>
<tr>
<td>Governance/control</td>
<td>23%</td>
<td>23%</td>
</tr>
<tr>
<td>Performance</td>
<td>17%</td>
<td>15%</td>
</tr>
</tbody>
</table>
Summary & Open Problems

**Summary:**
- Challenges: Operational visibility into complex cloud applications; need for real operational intelligence
- Opportunities: Transform systems to data; New line of ops data analytics; So many low-hanging pain points
- Agentless System Crawler and Vulnerability Advisor as simple ground-floor examples

**Parting Thoughts:**
- Operational Visibility >> Metrics & Logs (although a good start, add state, config, interactions, dependencies,...)
- Cloud lends itself to novel & elegant “monalytics” designed with cloud-native thinking
- Everything analytics can be as-a-service when we decouple systems | observations | recommendations | actions

**Open Research Questions:**
- Truly Seamless OpVis: No performance impact (~/~) + Absolutely no side effects (+/-)
- Scale out across runtimes and scale up to many instances; challenges & limits
- SW provenance, registry sprawl and analytics
- Privacy and data sensitivity with Ops data analytics
- Resource mgmt and accounting
- Piecemeal analytics/security solutions → Cloud analytics/security roadmap
- Visibility/Debuggability on the Fly
- Rules/annotators → Actually smart analytics that learn good and bad configs for security, performance, availability, etc.
- Cross-silo analytics across Time, Space, Dev/Ops [CloudSight Dream]
- AI 4 cloud & cloud 4 AI (i.e., NLP for Compliance; Mining for PSIRT)
Cloud Programming Models & Emerging Runtimes

Canturk Isci
IBM Research, NY
@canturkisci

Cloud Foundations 2
Bilkent Univ
Tue Aug 16, 10:30 AM
ABSTRACT: We are seeing an accelerating growth in cloud platforms, runtimes and programming models. Cloud discussion has shifted from utility and density to cloud-native services and design patterns. Emerging development, continuous integration and delivery techniques redefine how cloud applications are built with agility, quality and control. New cloud programming models raise the levels of compute abstraction to functions and high-level triggers.

In this talk I will present an overview of the emerging design patterns, programming models and the evolution of runtimes for cloud-native applications. We will continue from where we left off, containers and orchestration, and will discuss the design and delivery principles of cloud-native applications, focusing on DevOps and microservices. I will then present an overview of the emerging serverless computing model and its applications. I will highlight our current research activities and open-source innovations that both make use of these principles, as well as advance the state of the art in the field. I will conclude with some of the open problems and the opportunities to contribute.

Bio: Dr. Canturk Isci is a Research Manager and Master Inventor in IBM TJ Watson Research Center, Yorktown, US, where he leads the Cloud Monitoring, Operational and Security Analytics team. He currently works on deep introspection based monitoring techniques for cloud, and their application to novel operational, security and DevOps analytics. He is the technical lead for IBM Vulnerability Advisor for Containers and for Agentless System Crawler.

His research interests include operational visibility, analytics and security in cloud, virtualization, energy-efficient and adaptive computing. Prior to IBM Research, Dr. Isci was a Senior Member of Technical Staff at VMware, where he worked on distributed resource and power management. He has 50 academic papers and 30 issued or pending patents. Dr. Isci has a B.S. from Bilkent University, an M.Sc. with Distinction from University of Westminster, UK and a Ph.D. from Princeton University, US.
This Time

Cloud Foundations and Programming Models: From Infrastructure to ML/DL and Serverless

Deep Learning As a Service
(DLaaS, Watson ML)

Next-gen Cloud Infrastructure
(HW, Netw, DC, Accelerators,...)

Deep Learning As a Service
(DLaaS, Watson ML)

API Ecosystems
(API Harmony)

Operational Visibility and Analytics
(Deep Introspection, Crawlers and VA)

Microservices Framework
(Istio, Tracing, Analytics)

Cloud DevOps
(CI/CD, Pipelines, A/B, Canary, Toolchains)

Next-gen Cloud Platform
(Containers, Kubernetes, Netw)

Serverless Computing
(OpenWhisk)
Cloud Evolution: Trends – 4. Orch., Runtimes & PMs
Containers <3 Microservices
Some History

• Early distributed systems management
  – Keep system stable
  – Avoid accidental changes, root cause and fix problems
  – Converge to the stable state
  – Convergence and Promise Theory (CFEngine)

• Managing complex apps in cloud
  – Complex compositions and interactions, hard to converge to one stable state
  – Speed, ease of deployment and resource allocation
  – Quick response to market needs, rapid in-market experimentation [Agility] (Netflix, Etsy, etc.)
  – Continuous iterations and delivery, independent progress, design for failure
  – DevOps & Microservices
Agility with Control

**Agility:** In market experimentation with *speed*

**Control:** Ensuring *Compliance, Security, Resiliency* across DevOps flow

"Change is the only constant"
Microservices

- Small, independent services
- Access via well-defined interfaces (REST)
- Loosely coupled, dynamic binding, service registry and discovery
- Dynamic, externalized configuration
- Manage your own state, recovery, scaling
- Use data svcs for persistence
- Assume failure, build HA and recovery
- Scalable, reusable, simpler
- Resiliency patterns
  (Chaos (Monkey) Resistant
- Less cycle-efficient, as usual
  (asm > C > JVM > VM > Us)
Microservices and Dev[Sec]Ops Analytics

Agility with control and compliance

Pipeline (with compliance)
Vulnerability Advisor
Microservices and Dev[Sec]Ops Analytics

Agility with control and compliance

Pipeline (with compliance) → Red/Black Deploy → Vulnerability Advisor → 

Initial → Ramp-up (Phase) → Test (Phase) → Ramp-down (Phase)

Operationally 4 stages are needed

- Red: Routed traffic
- Black: No traffic
Microservices and Dev[Sec]Ops Analytics

Agility with control and compliance

1. Canary Rollout

- Time
- Old
- New

2. A/B Testing

- All traffic
- Same users see Version A
- Same users see Version B
- V1
- V2

A better than B?
Microservices and Dev[Sec]Ops Analytics

Agility with control and compliance

- Pipeline (with compliance)
- Red/Black Deploy
- Vulnerability Advisor

Risk-based vulnerability assessment for your organizations.

Worst 5 Images at Risk

- Worst 5 Images at Risk:
  - ibmserviceproxytenant: Risk Level 8
  - mying-fl: Risk Level 5
  - ibm-mobilefirst-starter: Risk Level 3
  - ibm-chaos-monkey: Risk Level 2
  - ibm-node-strong-pm: Risk Level 1

Top 5 Highest Risk Vulnerabilities

- Top 5 Highest Risk Vulnerabilities:
  - CVE-2015-7647: Risk Level 9
  - CVE-2013-7422: Risk Level 8
  - CVE-2015-8605: Risk Level 7
  - CVE-2016-2037: Risk Level 6
  - CVE-2016-0785: Risk Level 5

Top 5 Highest Impact Vulnerabilities

- Top 5 Highest Impact Vulnerabilities:
  - CVE-2013-7422
  - CVE-2016-2037
  - CVE-2015-7647
  - CVE-2016-0785
  - CVE-2016-0798
  - CVE-2015-1197
  - CVE-2016-0799
  - CVE-2015-0732
  - CVE-2014-4330

Number of Related Images: 2, 3, 3, 3, 3
Microservices and Service Meshes

Observation:
Microservices interact only over network using HTTP(s)

Insight:
SDN approach at L7 for visibility & control into comms between microservices

What:
- Service Registration
- Service Discovery
- Intelligent Routing

How:
- “Sidecar”
- Programmable L7 proxy
- Attached to each microservice
## Compute Evolution $\sim \Sigma$(Accidental Revolutions)

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel’s Accidental Revolution</td>
<td>4004 for Calculators $\rightarrow$ x86 GPPs</td>
<td>1970</td>
</tr>
<tr>
<td>Transmeta</td>
<td>X86 compatible Low-power processors $\rightarrow$ BT for x86</td>
<td>2000</td>
</tr>
<tr>
<td>VMware</td>
<td>Win on NIX with BT $\rightarrow$ Server virtualization, drives GPP virt (Vt-x,etc.)</td>
<td>2002</td>
</tr>
<tr>
<td>Xen, KVM</td>
<td>Commoditizing virtualization $\rightarrow$ Paves the way for cloud</td>
<td>2004</td>
</tr>
<tr>
<td>AWS</td>
<td>Bookstore/Web Svcs w SOI obsession $\rightarrow$ IaaS for the world</td>
<td>2007</td>
</tr>
<tr>
<td>CloudStack, OpenStack, etc.</td>
<td>Commoditizing cloud/IaaS $\rightarrow$ cloud platforms</td>
<td>2010</td>
</tr>
<tr>
<td>Docker (DotCloud)</td>
<td>Dev focused hosting svc $\rightarrow$ New way of SW delivery $\rightarrow$ container cloud</td>
<td>2013</td>
</tr>
<tr>
<td>Kubernetes (Google)</td>
<td>Warehouse computer $\rightarrow$ lmctfy,Borg, Omega $\rightarrow$ K8s cloud orchestration</td>
<td>2015</td>
</tr>
<tr>
<td>Serverless (AWS, IBM, MS, G)</td>
<td>FaaS/IFTT/Fipelines $\rightarrow$ Serverless, pay-per-use compute</td>
<td>2017</td>
</tr>
</tbody>
</table>
Challenge: Expensive to run microservices with traditional compute models

Break-down into microservices, e.g. one container per microservice

Make each micro service HA

Protect against regional outage

Explosion of # of containers / processes!

→ Increase of infrastructure cost footprint

→ Increase of operational mgmt cost & complexity
Answer: Serverless Execution model

- Scales inherently
  - One process per request
- No cost overhead for resiliency
  - No long running process to be made HA / multi-region
- Introduces event programming model
- Charges only for what is used
- Only worry about code → dev velocity, lower operational costs

![Diagram of Serverless Execution model]

Deploy action within millisecs, run it, free up resources
What is Serverless?

- a cloud-native platform for short-running, stateless computation and event-driven applications which scales up and down instantly, automatically, and transparently and charges at a millisecond granularity.
In a nutshell

Serverless (aka Functions-aaS) = consuming compute on a per-request basis
Analogy: Serverless is for compute what Object Storage is for storage: Consumption vs pre-allocation

**Object Storage**
- scales inherently
- no need to order capacity
- abstracts storage
  - user objects not disk blocks
- pay for what you use
  - per per object size vs allocated capacity

**Serverless**
- scales inherently
- no need to order capacity
- abstracts compute
  - Per-request execution of code
- pay for what you use
  - pay per request vs allocated capacity
Programming model

• Services define the events they emit as **triggers**, and developers associate the **actions** to handle the events via **rules**
  – Actions: JSON as in- and output

• The developer only needs to care about implementing the desired application logic - the system handles the rest
OpenWhisk overall architecture

REST

CLI

UI

iOS SDK

API Gateway*

CRUD triggers, actions, and rules
Invoke actions

Package
Feed

Package
Feed

Package
Feed

Package
Feed

Trigger

Rule

Action
NodeJS

Action
Swift

Action
NodeJS

Action
Docker

Rule

Rule

Rule

Action
Docker

Docker (and potentially other abstractions going forward)

Service ecosytem
- Bluemix services
- 3rd party services
- Self-enabled services

* work in progress
Summary & Open Problems

**Summary:**
- Emerging new modalities for compute, runtimes and programming models
- Tectonic shift from convergence and stability to breakneck agility and design for failure
  Chaos monkeys welcome >> needed
- New control and observation points; pave the way to intelligent, data-driven cloud ops

**Opportunities:**
- Many new observation points and control knobs across dev-ops
- Shift-left anything security, compliance, integrity, learning
- Systems and HW parallels exist and not exploited
- Many low hanging fruits for learning good and bad sw/system configurations
- Framework for self optimizing cloud applications
- Serverless is at its infancy; HW-SW codesign; Programming models for serverless
**Summary & Open Problems**

- **Open Research Questions:**
  - Obvious, AI 4 DevOps & DevOps 4 AI :)  
    Apply learning to understand config and behavior space of apps;
  - Cross-silo analytics across Time, Space, Dev/Ops [CloudSight Dream]  
    Work across and correlate Dev, Ops, Sidecar, System state data for operational insight
  - Deep visibility across DevOps cycle; Distributed tracing + Crawlers
  - Identifying key features: Configurations, KPIs, measures of goodness and gradations of failure
  - Blast-radius problems; Surviving partial failures; Self-healing; Maybe back to *Autonomic Systems*
  - Adversarial games; Microservice wars; Serverless trojan actions; Design for failure without trust
  - Resemblance to fault detection and fault grading; observability and controllability

- **From Peers:**
  - Optimization and scalability of the serverless compute core
  - Serverless programming models and tooling that address gaps in the development workflow
  - Breadth of solutions that can be addressed by serverless: IoT/Edge and hybrid
  - Also, any hackers are welcome to participate via our Apache incubator [OpenWhisk]
  - DevOps analytics use cases and learning based solutions
  - Contributors to Istio [Istio]
  - Applying deep learning skills to cloud problems, and systems, infra skills for Deep Learning on Cloud
Learn more: Open Innovation <3

Agentless System Crawler
Web: https://www.google.com.tr/search?q=%22agentless+system+crawler%22
Twitter: https://twitter.com/ibmbluemix & https://twitter.com/canturkisci
DwOpen: https://developer.ibm.com/open/agentless-system-crawler/
Git: https://github.com/cloudviz/agentless-system-crawler
DockerHub: https://hub.docker.com/u/cloudviz/
Run: http://console.ng.bluemix.net/
SlideShare: http://www.slideshare.net/canturkisci/
YouTube: https://www.youtube.com/channel/UCf8Fn8dKQzBCJRgIr1jOlGYg
Podcast: https://soundcloud.com/thenewstackmakers/creating-analytics-driven-solutions-for-operational-visibility
Pubs: http://canturkisci.com/ETC/MYpublications.html

Vulnerability Advisor
Web: https://www.google.com.tr/search?q=%22vulnerability+advisor%22+ibm
Twitter: https://twitter.com/ibmbluemix & https://twitter.com/canturkisci
Run: http://console.ng.bluemix.net/
SlideShare: http://www.slideshare.net/canturkisci/
YouTube: https://www.youtube.com/channel/UCf8Fn8dKQzBCJRgIr1jOlGYg
Pubs: http://canturkisci.com/ETC/MYpublications.html
Learn more: Open Innovation <3

DevOps and Microservices (Amalgam8 & Istio)
- Web: [https://www.google.com.tr/search?q=%22istio%22](https://www.google.com.tr/search?q=%22istio%22)
- Twitter: [https://twitter.com/ibmbluemix](https://twitter.com/ibmbluemix)
- io: [https://istio.io/](https://istio.io/)
- Git: [https://github.com/amalgam8/amalgam8](https://github.com/amalgam8/amalgam8)
- Git: [https://github.com/istio/istio](https://github.com/istio/istio)

Serverless (OpenWhisk)
- Web: [https://www.google.com.tr/search?q=%22openwhisk%22+ibm](https://www.google.com.tr/search?q=%22openwhisk%22+ibm)
- Twitter: [https://twitter.com/openwhisk](https://twitter.com/openwhisk)
- Git: [https://github.com/openwhisk/openwhisk/](https://github.com/openwhisk/openwhisk/)
- Slack: [https://dwopen.slack.com](https://dwopen.slack.com) (channel: openwhisk)
- SlideShare: [http://www.slideshare.net/OpenWhisk](http://www.slideshare.net/OpenWhisk)
- YouTube: [https://www.youtube.com/channel/UCbzgShnQk8F43NKsvEYA1SA](https://www.youtube.com/channel/UCbzgShnQk8F43NKsvEYA1SA)
Thank You

Operational Visibility and Analytics Designed for Cloud
[feat. Agentless System Crawler & Vulnerability Advisor]

Cloud Programming Models & Emerging Runtimes
[feat. istio & OpenWhisk]

IBM Research
Cloud Monitoring, Operational and Security Analytics

http://www.canturkisci.com/blog
@canturkisci