



Runtime Control

Orchestration of Cloud Resources

- Kubernetes orchestrates containers
- Other orchestrators may control different resources
 - **Aether** controls quality of service to users allocating radio resources
 - **Cisco's Meraki** controls Wi-Fi access points
 - **GCE/EC2/Azure** control servers and network devices hosting VMs

Main tasks in runtime control

- Authenticate the actor wanting to perform an operation
- Check whether the actor has sufficient privileges
- Push new configuration to back-end components
- Store new configuration in a persistent database

Low-level configuration is required

- **Aether** controls quality of service to users allocating radio resources
 - Configure radios to allocate bandwidth to users
 - Configure switches and routers correctly route and prioritize packets
- **Cisco's Meraki** controls Wi-Fi access points
 - Configure radios to avoid interference between access points
 - Configure authentication services (e.g., WPA2) on the access points
 - Configure switches and routers in the wired back-end
- **GCE/EC2/Azure** control servers and network devices hosting VMs
 - Configure and update servers
 - Configure hypervisor to create new VMs and volumes
 - Configure (virtual) switches to manage virtual networks

High-level abstractions even more important

- Need to create an abstraction layer on top of back-end components
- Define APIs to manipulate the abstractions
- Back-end components may be individual physical devices or software, but could also be groups of related components

Challenges in building abstractions

- Level of abstraction and visibility over implementation details



VS



VS



Challenges in building abstractions

- Level of abstraction and visibility over implementation details



VS



VS



- Access control granularity and identifying actors

<input type="checkbox"/>		italocunha@gmail.com	Ítalo Cunha	BigQuery Data Viewer	14/16 excess permissions	▼
				BigQuery Job User	3/4 excess permissions	▼
				Viewer	2484/2540 excess permissions	▼
<input type="checkbox"/>		jh4001@columbia.edu	Jia He	BigQuery Admin	103/104 excess permissions	▼
				BigQuery Data Editor	35/36 excess permissions	▼
				Compute Admin	597/599 excess permissions	▼
				Compute Image User	7/9 excess permissions	▼
				Service Account User	4/5 excess permissions	▼
				Storage Transfer User	19/20 excess permissions	▼
				Viewer	2535/2540 excess permissions	▼
<input type="checkbox"/>		ls3748@columbia.edu	Loqman Salamatian	BigQuery Admin	104/104 excess permissions	▼
<input type="checkbox"/>		lw2863@columbia.edu		BigQuery Data Viewer	16/16 excess permissions	▼

Challenges in building abstractions

- Level of abstraction and visibility over implementation details



VS



VS



- Access control granularity and identifying actors

<input type="checkbox"/>		italocunha@gmail.com	Ítalo Cunha	BigQuery Data Viewer
				BigQuery Job User
				Viewer
<input type="checkbox"/>		jh4001@columbia.edu	Jia He	BigQuery Admin
				BigQuery Data Editor
				Compute Admin
				Compute Image User
				Service Account User
				Storage Transfer User
				Viewer
<input type="checkbox"/>		ls3748@columbia.edu	Loqman Salamatian	BigQuery Admin
<input type="checkbox"/>		lw2863@columbia.edu		BigQuery Data Viewer

Permissions for italocunha@gmail.com

Current permissions for Viewer role

Last Analyzed 6/20/22	1 appengine.applications.get
	2 appengine.services.list
	3 bigquery.datasets.get
	4 billing.resourceCosts.get
	5 clientauthconfig.clients.list
	6 cloudnotifications.activities.list
	7 cloudsql.instances.export
	8 cloudsql.instances.list
	9 cloudtrace.insights.list
	10 cloudtrace.tasks.list
	11 cloudtrace.traces.list
	12 compute.addresses.list
	13 compute.autoscalers.list
	14 compute.backendServices.list
	15 compute.disks.list
	16 compute.diskTypes.list
	17 compute.firewalls.list
	18 < [redacted] >

16/16 excess permissions

Challenges in building abstractions

- Level of abstraction and visibility over implementation details



VS



VS



- Access control granularity and identifying actors
- Fuzzy terminology (as in “orchestration” vs “runtime control”)

Abstractions depend on service

- In the case of a cloud, the usual culprits apply
 - Bare-metal servers, virtual machines, containers
 - Virtual disks, database services, key-value stores, distributed file systems
 - Virtual networks, virtual interfaces, firewalls, load balancers
- PaaS need abstractions for the services they provide
 - Heroku (Web applications) vs mPaaS (mobile PaaS)
- In Aether
 - Enterprises: sites, facilities, cells
 - Slices: device groups, apps, spectrum allocation
 - Traffic classes: link quality, resource reservations

Models and state

- A runtime control system needs to maintain (desired) state
 - Persistent storage
 - The configuration of the runtime control system is authoritative
 - Need to be available and backed-up
 - Replication, consensus systems, and possibly stored in a repository
 - Versioning and migration
 - Configuration should be versioned in case rollback is necessary
 - Also allows execution of multiple versions of the runtime control system
 - Synchronization
 - If some component fails, its state should be recovered from the runtime control system after failure restoration

Runtime state

- Runtime control also needs to store the *current* state of the system
 - Not something that we store on repositories
 - Failed servers, ongoing issues, network state
 - Outside of GitOps
 - Key-value store or some other reliable storage database
 - Need redundancy against crashes or state of the system would be lost

Runtime Control API

- Need to expose control over infrastructure
 - Runtime state is incompatible with GitOps, so need an API

Runtime Control API

- Need to expose control over infrastructure
 - Runtime state is incompatible with GitOps, so need an API
- Flexible operations to support tool development
 - Like in a rest API (get, put/post, patch, delete)
- Parameter validation and error reporting
 - Ease of use, user friendliness
 - Prevent database corruption or inconsistent states
- Access control and logging
 - Post-mortem audits of who or what performed each action

Runtime Control API maintenance

- Can automatically generate the API from the configuration models
 - **Pros:** Less code to write, guaranteed consistency
 - **Cons:** Any changes to the models will immediately materialize in the API
 - Harder to maintain backwards compatibility

Aether Runtime Control

