AWS re:Invent

GPSTEC337

Architecting multi-tenant PaaS offerings with Amazon EKS

Judah Bernstein
Sr. Partner Solutions Architect, SaaS Factory
Amazon Web Services

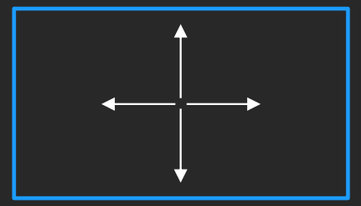
Ranjith Raman
Sr. Partner Solutions Architect, SaaS Factory
Amazon Web Services

AWS re:Invent

aws

What is Kubernetes?

Open-source container management platform



Helps you run containers at scale

Gives you primitives for building modern applications

Platform as a Service (PaaS)

What is Platform as a Service (PaaS)?

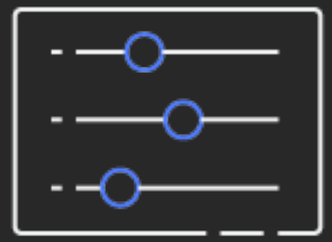
Platform as a service (PaaS) is a proven model for running applications without the hassle of maintaining on-premises hardware and software infrastructure at your company

https://www.salesforce.com/ap/paas/overview/

Core requirements for today's PaaS solution

External customers

Custom applications



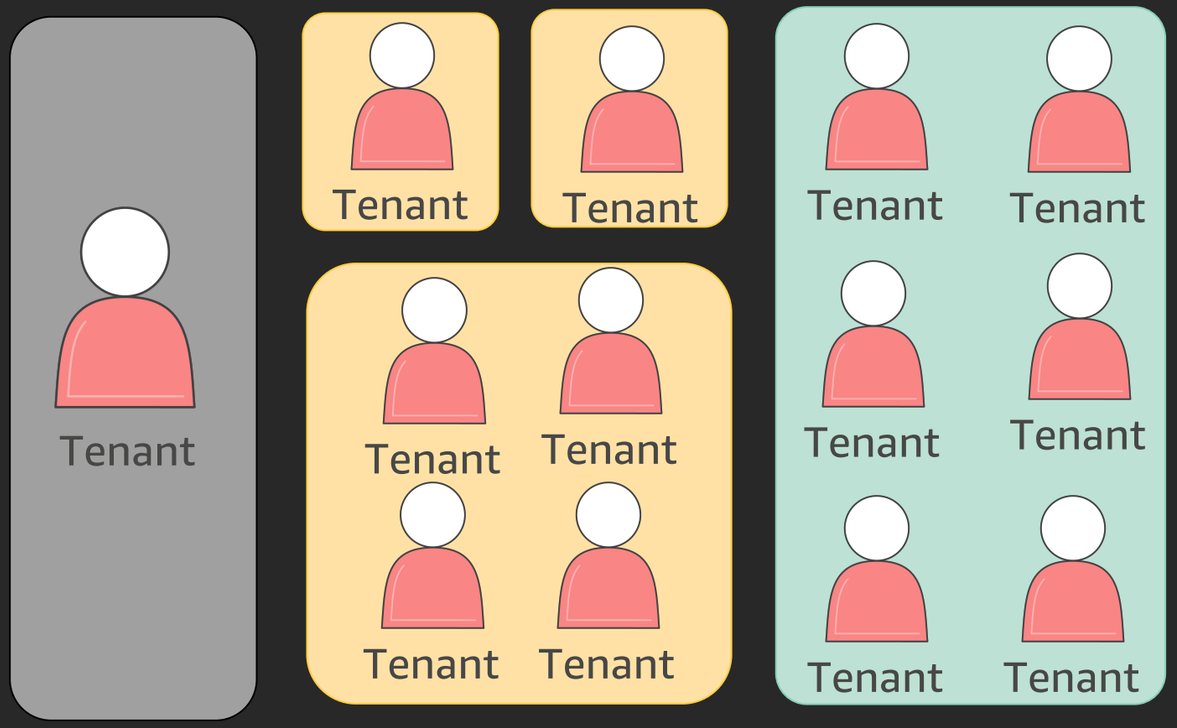
Automatic scaling

Turnkey deployment

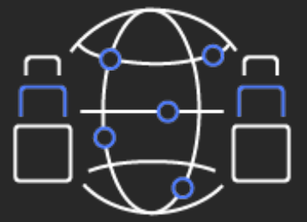
Multiple languages

Secure & compliant

Tenant isolation design considerations



Compute isolation



Network isolation

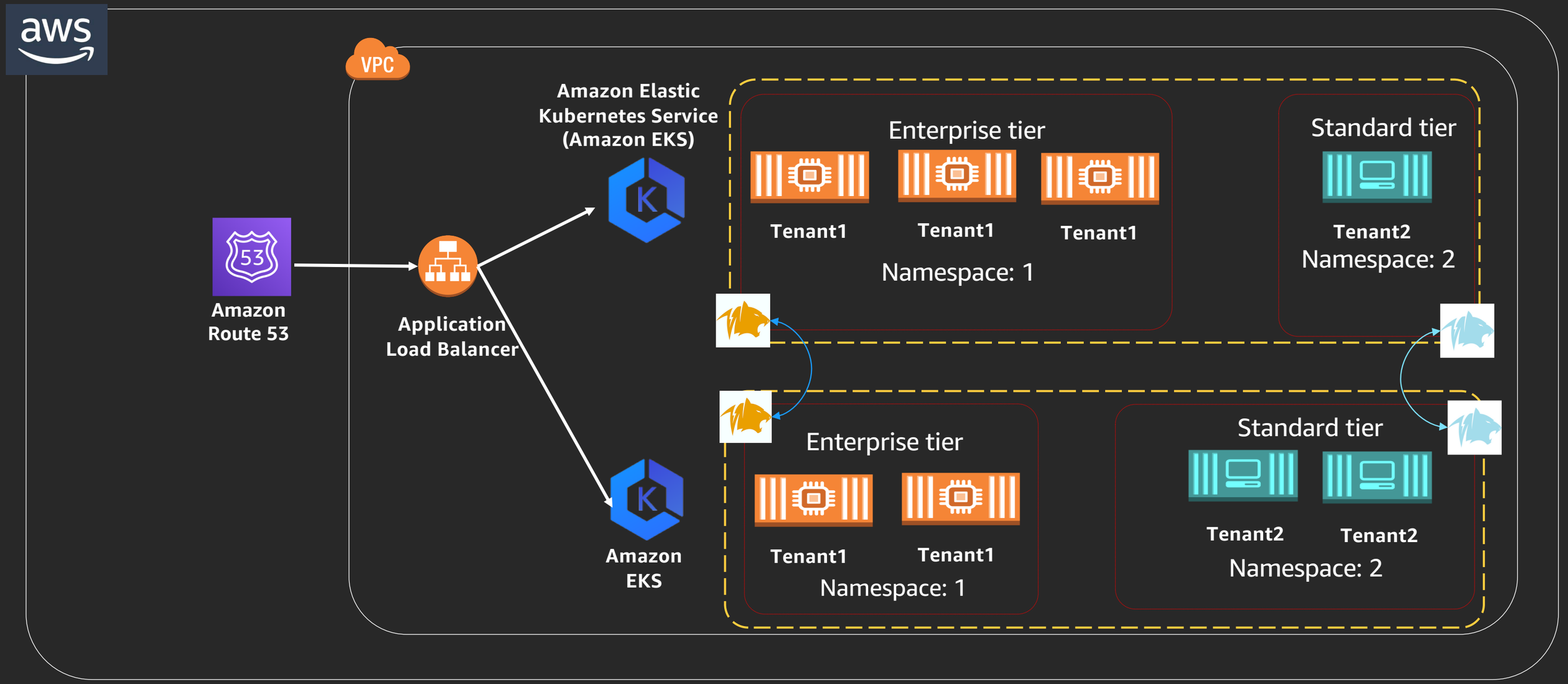
Plan tiering

Storage isolation

Usage metering

Advanced architectures

Multi-tenant Kubernetes workload architecture



Kubernetes tenant isolation strategies

Native multi-tenancy

The Kubernetes open-source project
DOES NOT currently support
native multi-tenancy

Soft multi-tenancy

- Hospitable tenants
- Tenants == business units
- No intention to exploit
- Accident prevention
- Focus on agility

Hard multi-tenancy

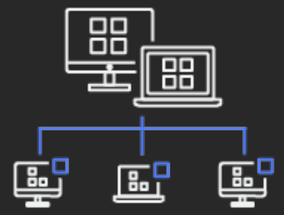
- Inhospitable tenants
- Tenants == different companies
- Potential intention to exploit
- Focus on tenant isolation

PaaS architecture requirements

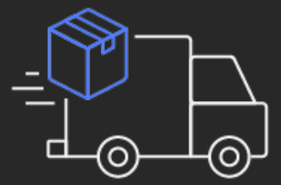
Core components of a PaaS

API wrapper

Workload manager



Customer endpoint

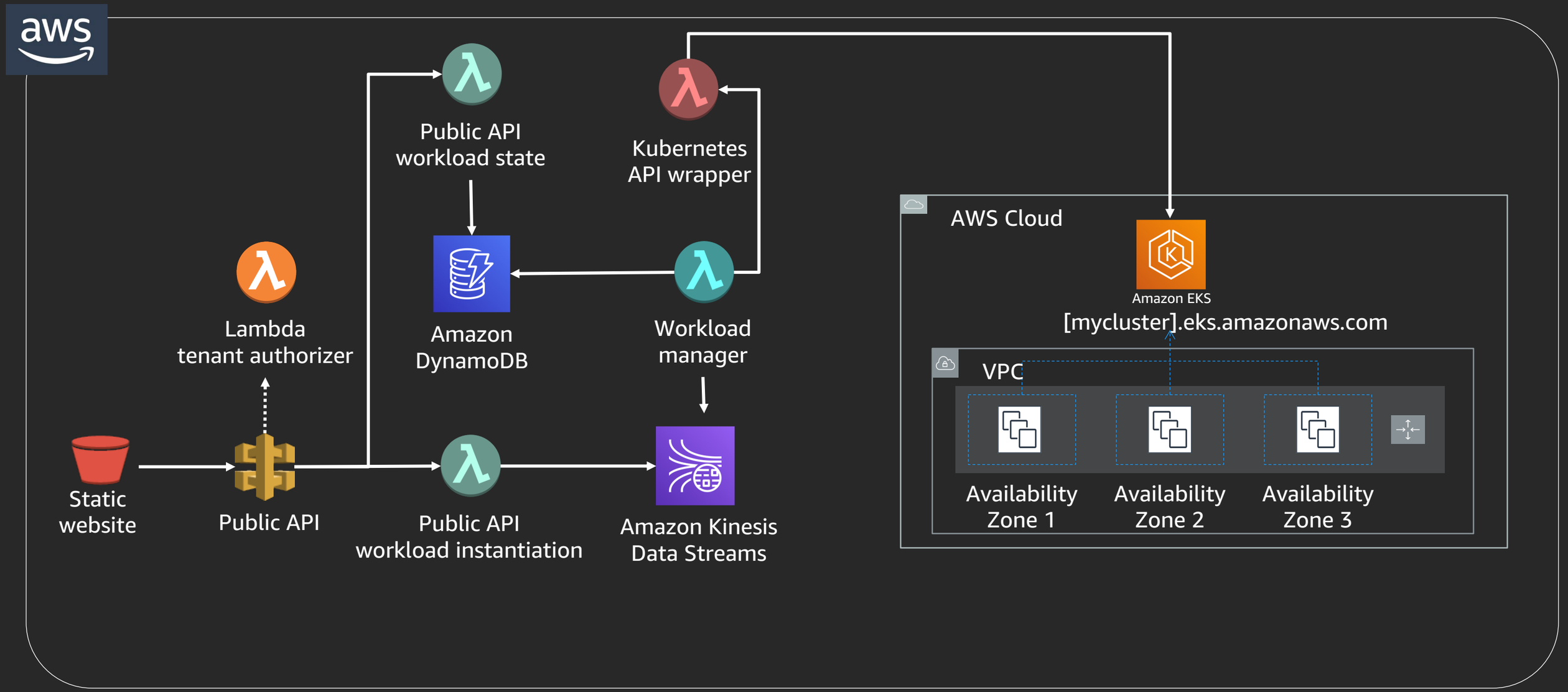


Deployment pipeline

Monitoring service

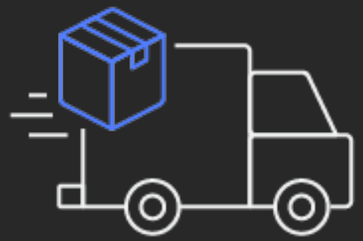
Metering service

Multi-tenant PaaS API layer



Compute isolation

Design considerations for compute isolation



Namespace isolation

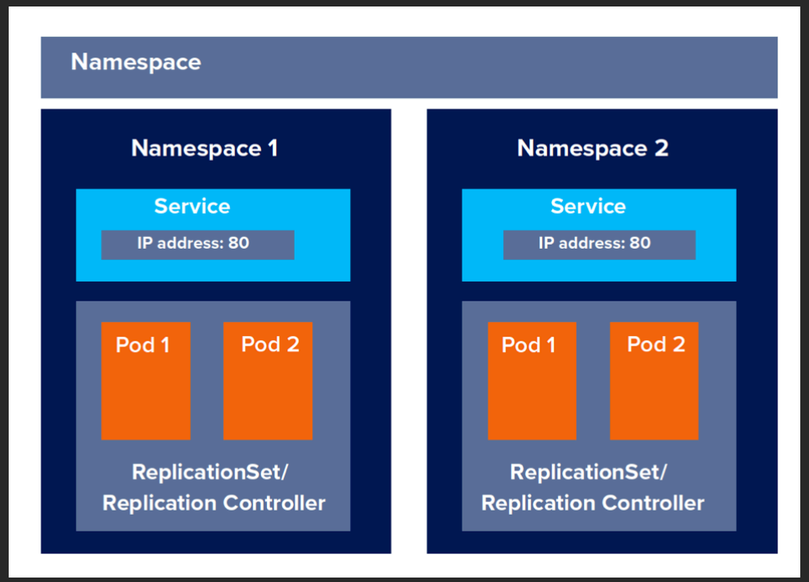
ServiceAccount segmentation

Resource isolation

Namespace per tenant

Namespaces

Provide a scope for names. Names of resources need to be unique within a namespace but not across namespaces. Namespaces cannot be nested inside one another, and each Kubernetes resource can only be in one namespace.



Strength

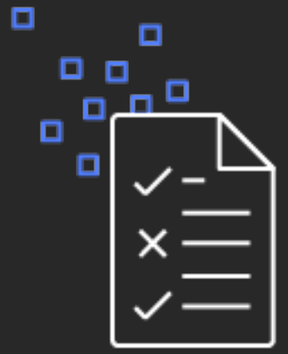
Provides obfuscation of tenant resources within a pooled environment by mitigating resource advertisement

Weakness

You can still globally access tenant resources in a pooled environment

https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/
https://www.weave.works/blog/optimizing-cluster-resources-for-kubernetes-team-development

Securing platform as a service authorization



Restrict role access

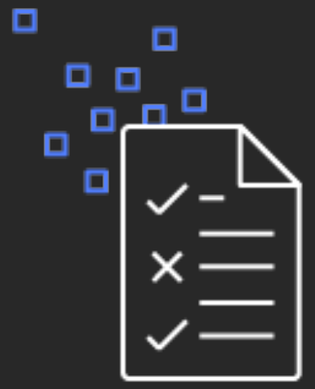
Tenant service accounts

Tenant RoleBinding

Example - Role, ServiceAccount, RoleBinding

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: tenant1-role-binding
  namespace: tenant1
subjects:
- kind: ServiceAccount
  name: tenant1-service1
  namespace: tenant1
roleRef:
  kind: Role
  name: tenant1-role
  apiGroup: ""
```

Restricting access to Kubernetes compute



Pod security policies

A set of conditions that pods must meet to be accepted by the cluster

DISABLE PRIVILIGED MODE!!!

Restrict deployment of tenant pods from accessing

- EC2 Instance
- Filesystems
- Networks
- Processes (PID)
- Namespaces
- Volumes

Preventing a tenant's custom application from accessing another tenant's resources

Example – Pod security policy

```
apiVersion: extensions/v1beta1
kind: PodSecurityPolicy
metadata:
  name: pod-security-policy
spec:
  privileged: false
  runAsUser:
    rule: MustRunAsNonRoot
  seLinux:
    rule: RunAsAny
  fsGroup:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  volumes:
  - '*'
```

Prevents creation of privileged containers

Prevents containers that require root privileges

Note!

Amazon EKS 1.13 cluster now has the pod security policy admission plugin enabled by default

Amazon EKS 1.13 and above

$ kubectl get psp

```
NAME              PRIV   CAPS   SELINUX    RUNASUSER   FSGROUP    SUPGROUP   READONLYROOTFS   VOLUMES
eks.privileged    true   *      RunAsAny   RunAsAny    RunAsAny   RunAsAny   false            *
```

$ kubectl describe psp eks.privileged

```
Name:  eks.privileged
Settings:
  Allow Privileged:                true
  Allow Privilege Escalation:      0xc0004ce5f8
  Default Add Capabilities:        <none>
  Required Drop Capabilities:      <none>
  Allowed Capabilities:            *
  Allowed Volume Types:            *
  Allow Host Network:              true
  Read Only Root Filesystem:       false
```

Create a more restrictive pod security policy and delete the default

Securing platform as a service authentication

Never expose ServiceAccount

Pull Secrets from vault

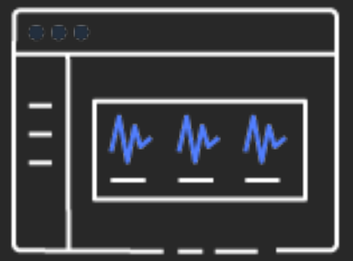
Consider federation

Network isolation

Design considerations for network isolation

Network &
pod policies

Service-
mesh



Monitoring &
compliance

Network isolation with Kubernetes

Network policy

- Define policy to control network traffic
- No in-built mechanism to enforce the policy
- Use network plugins
  - Tigera Calico and Secure
  - Weave, Romana
- Network ingress and egress rules

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: default-deny
  namespace: tenant1
spec:
  podSelector:
    matchLabels: {}
  types:
  - Ingress
  - Egress
```

Example - Network policies

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: allow-same-namespace
  namespace: tenant1
spec:
  podSelector:
    matchLabels:
      app: backend
  ingress:
  - from:
    - podSelector:
        app: frontend
```

Implementing a K8S tenant network policy

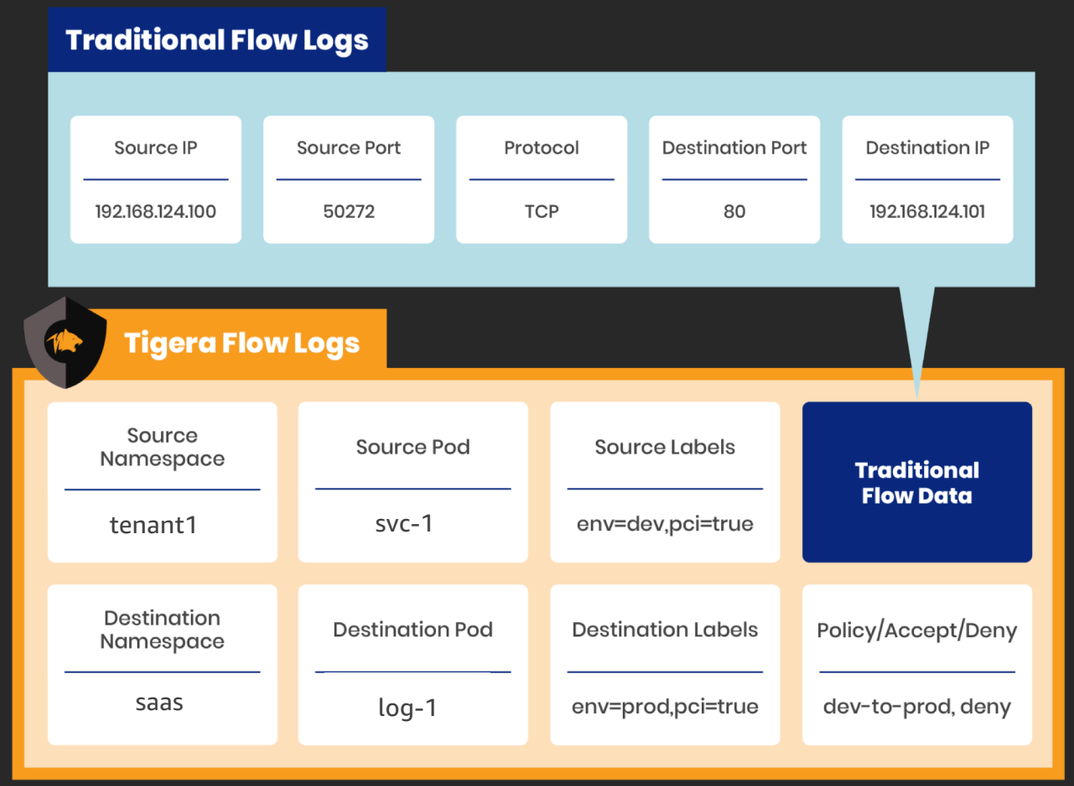
```
# Save the Following Tenant Namespace Policy
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: deny-from-other-namespaces
spec:
  podSelector:
```

```
# Apply Network Policy to Tenant1 and Tenant2
```

```
# Create a Cross Tenant Event "tenant2" => failure
curl http://nginx.tenant2
```

Tigera Secure: Network security for Kubernetes

- Least privileged tenant access through fine-grained network policies [L3-L7] (Calico)
- Enable tenant traceability through Amazon CloudWatch custom metrics and logs, and Prometheus
- In-transit encryption for all tenant communication
- FQDN support with whitelists and default-deny
- Automate incident response workflows for tenants
- Monitor, detect, prevent malicious tenant activity
- Continuous compliance (PCI DSS, SOC2, etc.)
- Immediate alerting for cross-tenant impacts
- Reporting, threat analysis, and logging dashboards



Atlassian achieves AWS multi-tenancy with TIGERA

"Atlassian's objective was to build a central, managed container platform, hosted in AWS, that could eventually support the majority of its compute workloads. That includes . . . Bitbucket/Bamboo code management and continuous integration and deployment (CI/CD) platform. This also builds and runs customers' code – i.e., arbitrary code execution within a multi-tenant environment: a security professional's worst nightmare!"

"Our security approach had to be strong enough to isolate not just our own developers' applications but, more importantly, our external customers' code.

Corey Johnston, Kubernetes Platform Senior Team Lead at Atlassian

"This led us to Tigera Calico as the most robust, Kubernetes-native network security solution for achieving microsegmentation of container workloads."

https://www.tigera.io/customer/atlassian

Create logical tenant network boundaries

Service mesh

A logical boundary for network traffic between the services that reside within it. A mesh can contain virtual services, virtual nodes, virtual routers, and routes.

AWS App Mesh

Istio

Consul

Linkerd

SuperGloo

- Abstraction
- Service discovery
- Obfuscation
- Retry logic
- Circuit breakers
- Routing
- Load balancing
- Security
- Observability

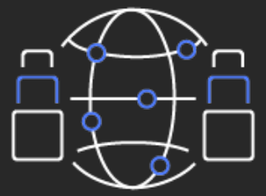
https://istio.io/blog/2018/soft-multitenancy/

Multi-tenant isolation with service mesh

Tenant routing

Network isolation

Obfuscation



Service discovery

Canary deployment

Multi-region

Tenant tiering & quality of service

Tenant tiering strategies with resource quotas

- Limit total resources consumed by a tenant
- Set a quota and resource limit for tenant (namespace) capacity
- Limit the type of resources consumed by a tenant
- Enforce performance and capacity tiering (CPU, mem, storage)
- Manage tenant pod prioritization
- Minimize blast radius for a tenant overconsumption event

https://kubernetes.io/docs/concepts/policy/resource-quotas/

Tenant tiering strategies with resource quotas

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: compute-resource
spec:
  hard:
    pods: “5”
    requests.cpu: "1"
    requests.memory: 2Gi
    limits.cpu: "2"
    limits.memory: 4Gi
```

https://kubernetes.io/docs/concepts/policy/resource-quotas/

Managing tenant placement on Kubernetes nodes

Node affinity (Tenant stickiness)

- Constrain which nodes a tenant’s pods are deployed to with labels and selectors
- Allows certain tenant pods to be scheduled on certain specified nodes

Pod anti-affinity (tenant silo)

- How tenant (pods) should be placed in relationship to where other tenants (pods) are deployed
- Prevents scheduling tenant (pod) deployment on the same nodes as pods of another tenant

Defining hard tenant node placement boundaries

Taints

- A *taint* lets you mark a node to prevent the scheduler from using it for certain pods

Tolerations

- A *toleration* lets you designate Pods that can be used on "tainted" nodes

```
kubectl taint nodes node1 client=tenant1:NoSchedule
```

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    env: test
spec:
  containers:
  - name: nginx
    image: nginx
    imagePullPolicy: IfNotPresent
tolerations:
- key: “client”
  operator: "Equal"
  value: “tenant1”
  effect: "NoSchedule"
```

https://kubernetes.io/docs/concepts/configuration/taint-and-toleration/

Managing eviction through priority and preemption

Priority

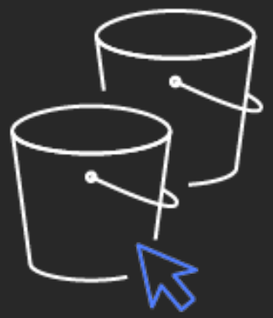
- Priority indicates the importance of a pod relative to other pods
- High priority pod placed in front of the queue
- Prioritize tenants base on plan tiers

Preemption

- Allows a tenant to be evicted, to prioritize another tenant
- Be mindful of how the priority values are assigned
- Graceful tenant task termination for preemption victims

Storage isolation

Design considerations for storage multi-tenancy



Folder isolation

Volume isolation

AWS resource isolation

Storage isolation design considerations

- Artifact roles limit the physical volumes a tenant “namespace” can access
- PVC Claims limit a tenant’s access to NFS volume subfolders through RBAC
- Storage classes exist at the cluster level (not namespace)
- Use storage resource quotas to limit by namespace
- To deny a specific namespace access to specific storage, set the resource quota to 0 for that storage class

Native AWS IAM roles for service accounts, pods

- IAM Roles for ServiceAccounts (IRSA)
- Pods first class citizens in IAM
- AWS identity APIs recognize Kubernetes pods
- OpenID Connect (OIDC) IDP with a Kubernetes service account annotations allow you to use IAM roles at the pod level.
- Exchange OIDC JWT with STS for temporary credentials
- Signed token verified by STS with OIDC Provider
- Hosted on the control plane and managed by AWS

https://aws.amazon.com/blogs/opensource/introducing-fine-grained-iam-roles-service-accounts/

Example of an AWS IAM role for a service account

kubectl get sa tenant1-serviceaccount1 -o yaml

```
apiVersion: v1
kind: ServiceAccount
metadata:
  annotations:
    eks.amazonaws.com/role-arn: arn:aws:iam::123456789012:role/Tenant1Role
  name: tenant1-serviceaccount1
  namespace: default
secrets:
- name: tenant1-mysecret
```

Designing for cost

Optimizing multi-tenant Kubernetes for cost

- Resource aggregation to improve utilization and performance across pool of tenants
- Meter cluster usage (cpu, mem, storage) for each tenant
- Develop tiering strategies, and provider budgets per plan type
- Develop performance and tenant SLA tiering
- Use metered information to drive entitlement, showback, chargeback, and compliance
- Align cost structure to your customer’s consumption
- Tie consumption to native K8S constructs such as Pods, PersistantVolumes

Metering Kubernetes cost per tenant

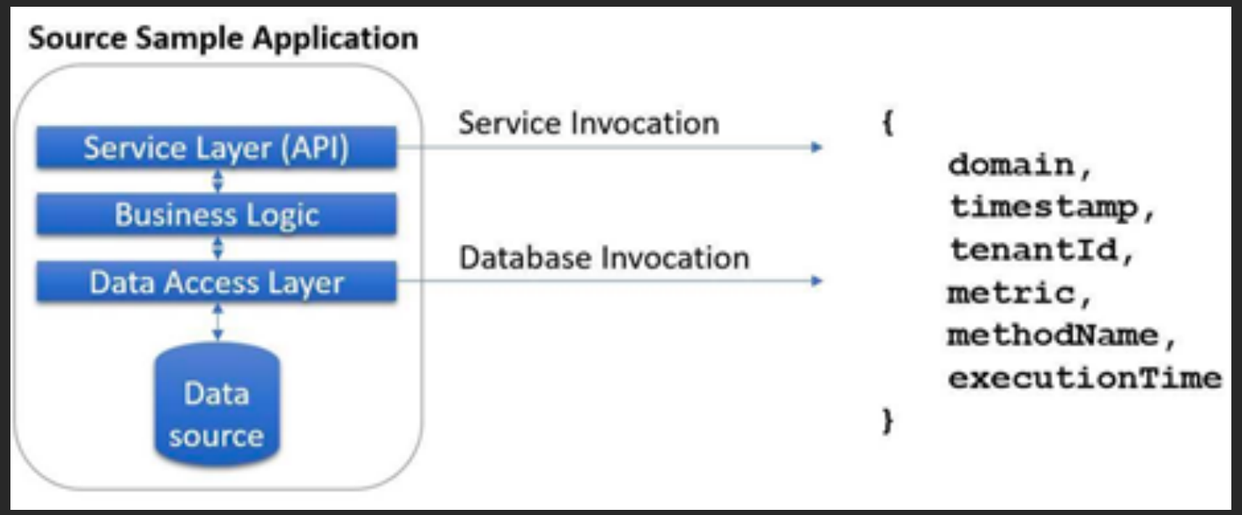
- Expose Kube State Metrics for tenant (namespace) metrics
    - Number of pods per hour
    - Pod usage per hour (cpu, mem)
    - EC2 Node metrics (instance type)
- Lambda functions expose opaque, transparent metrics
- Meter events (start, stop), or regularly poll on schedule
- Pull tenant Metrics into real time analytics pipeline
- Digest metrics into unit based on consumption type
- Define API dimensions to post metering records
- Send records to a metering/billing API (schedule)

Amazon CloudWatch Container Insights

https://github.com/kubernetes/kube-state-metrics/tree/master/docs#exposed-metrics

Slalom solution approach for cost per tenant

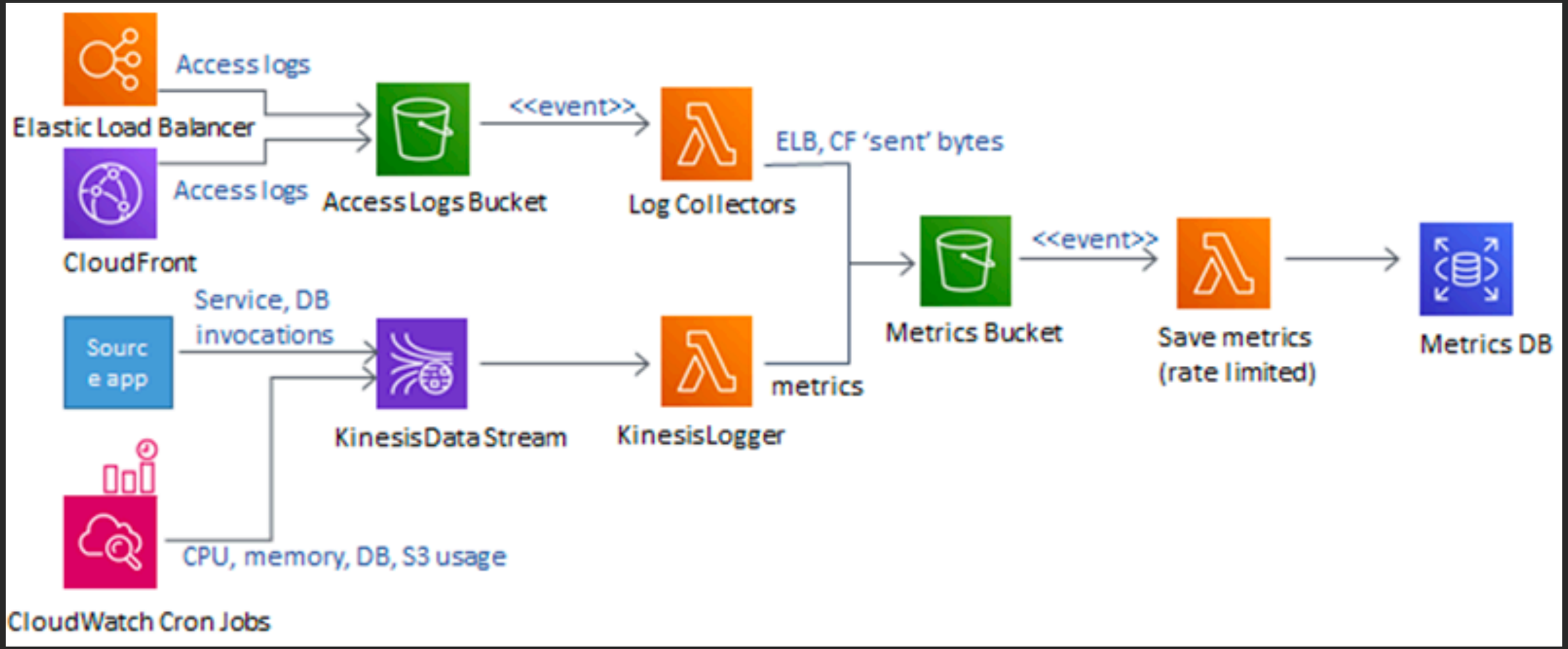
- Number of API requests processed, and execution time per use-case
- CPU and memory usage across the pods in the K8S tenant namespace
- Single tenant S3 bucket, CloudFront distribution



timestamp	tenant	metric	units
2019-05-05T06:30:49	tenant1	api_invocation	5
2019-05-05T06:30:49	tenant1	db_invocation	2
2019-05-05T06:30:49	tenant2	cpu	452549465
2019-05-05T06:30:49	tenant1	elb_bytes_sent	242
2019-05-05T06:30:49	tenant1	S3_storage	1024
2019-05-05T06:30:49	tenant2	memory	12267365
2019-05-05T06:30:49	tenant1	cf_bytes_sent	102
2019-05-05T06:30:49	tenant2	db_storage	4096

https://aws.amazon.com/blogs/apn/calculating-saas-cost-per-tenant-a-poc-implementation-in-an-aws-kubernetes-environment/ https://www.hawkular.org/hawkular-metrics/docs/user-guide/#_tenants

Slalom cost per tenant architecture



Advanced multi-tenant strategies

AWS re:Invent

aws

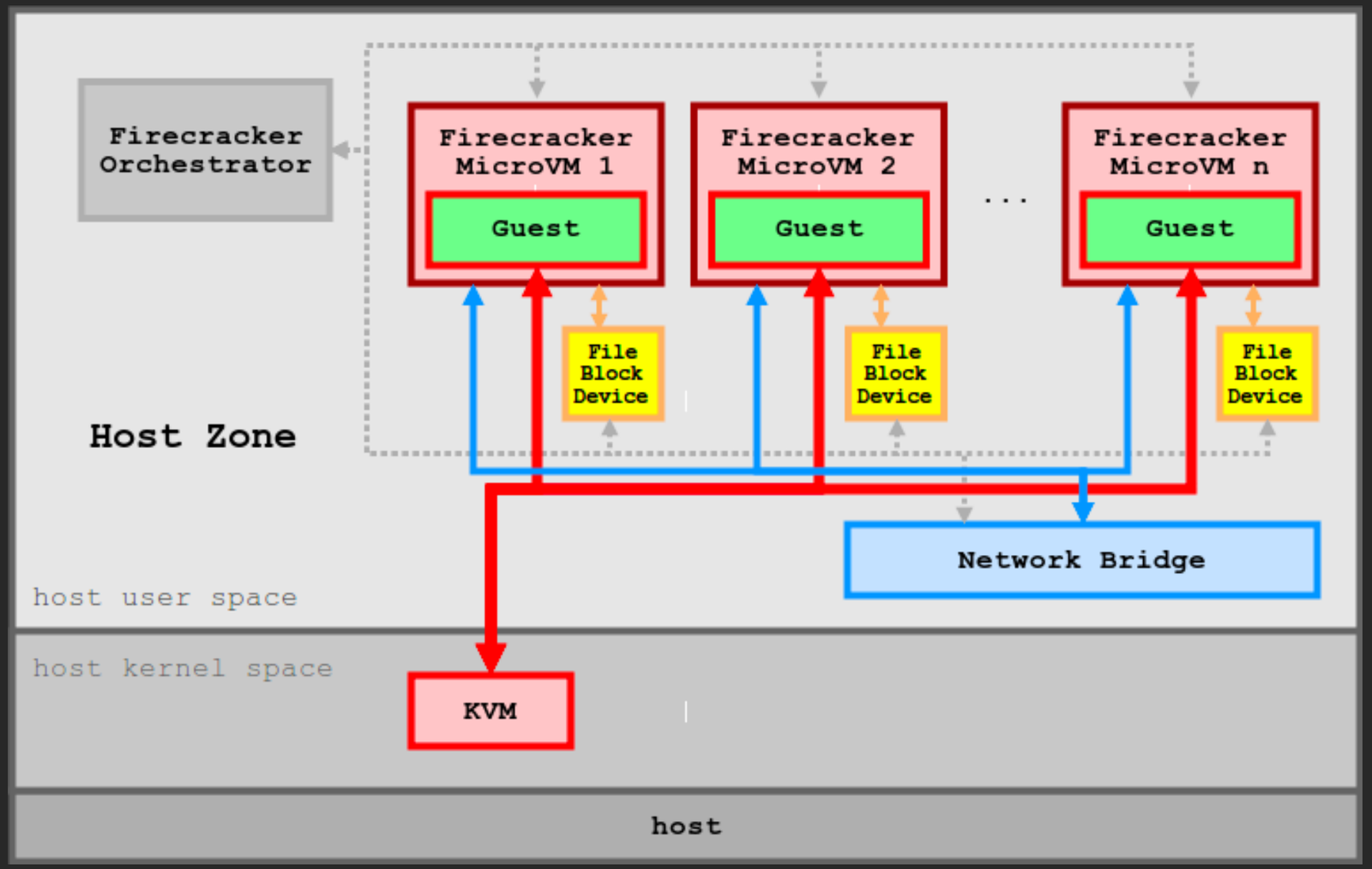
Hardware isolation

Hardware-isolation between containers giving you the highest level of security between the services in your cluster

Kata Containers is an open source project and global community working to build a standard implementation of lightweight virtual machines that feel and perform like containers

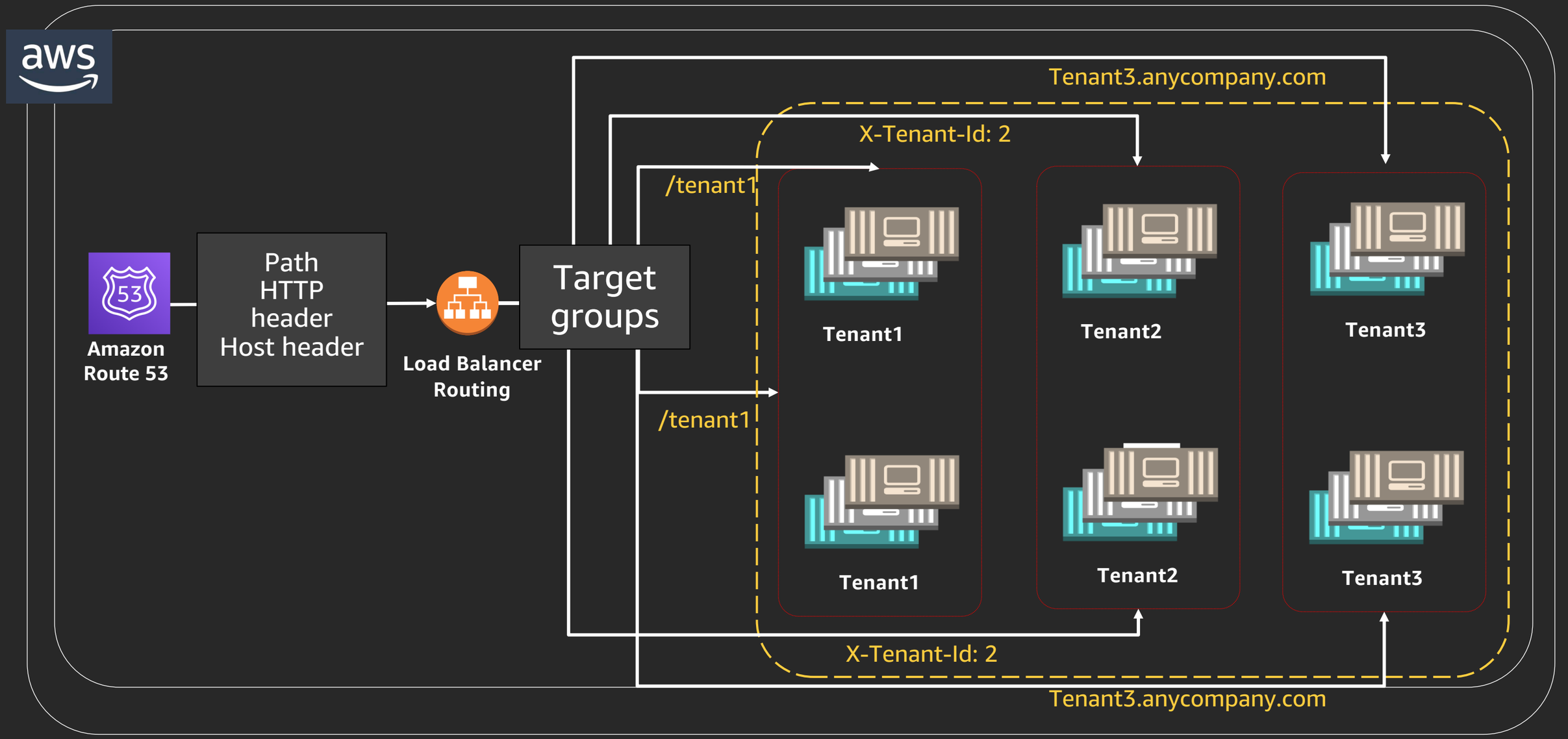
Firecracker is an open source virtual machine monitor (VMM) that uses the Linux Kernel-based Virtual Machine (KVM). Firecracker allows you to create micro virtual machines or microVMs

Multi-tenant MicroVM isolation architecture



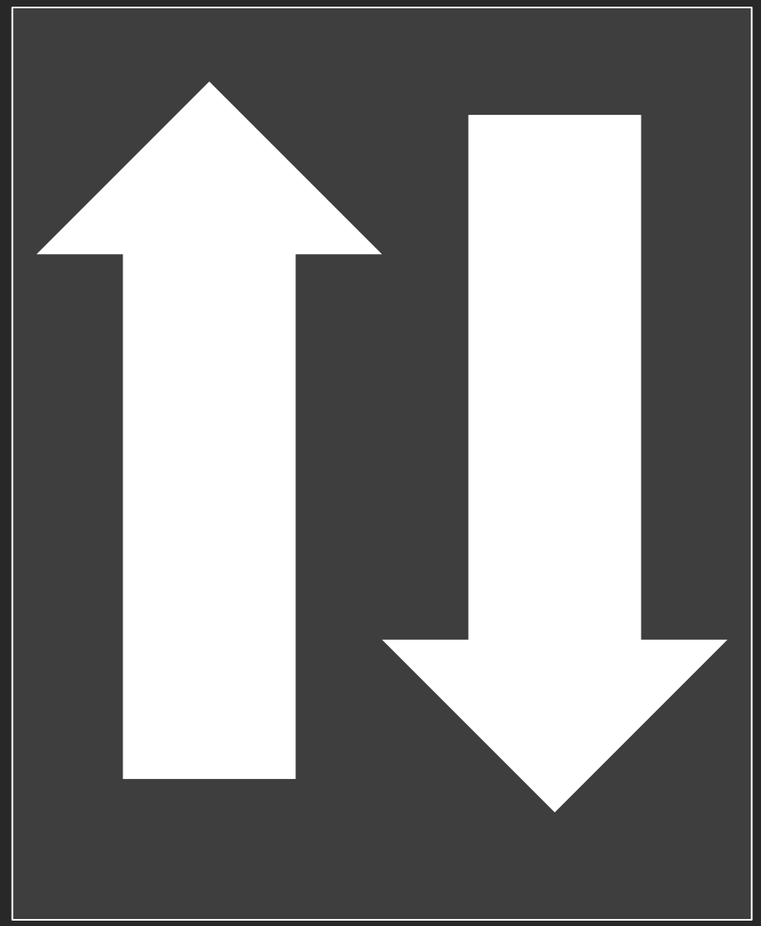
https://github.com/firecracker-microvm/firecracker/blob/master/docs/design.md

Application Load Balancer ingress controller V2

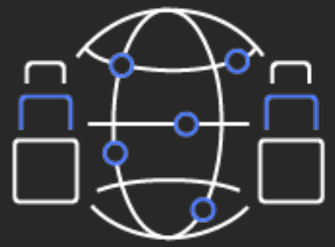


Vertical Pod Autoscaler for Database Isolation

- Automatically adjusts the CPU and memory reservations for pods to help right-size your applications
- This can free up CPU, memory for other pods for better resource allocation
- It can scale a database application to automatically support the necessary utilization required by the tenant consuming the database
- Dedicated Database Volume per tenant, multi-tenant database deployment



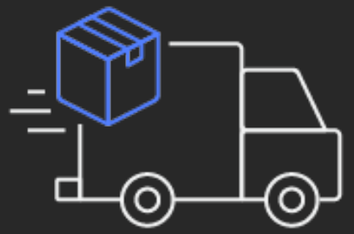
Things we did not have sufficient time to cover…



SaaS monitoring

Emerging K8s concepts

Image scanning



SaaS devops

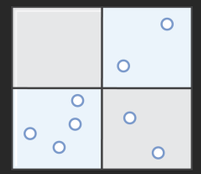
And so much more…

Closing thoughts

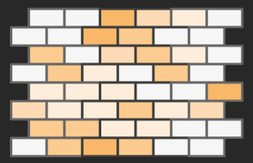
Things to remember

No native support

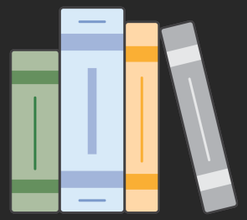
Do it yourself (DIY)



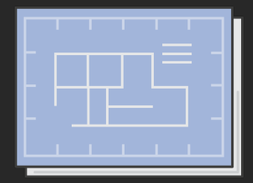
Community



Never expose API



Deep dive on K8S



Silo pipeline

Related sessions

Related SaaS sessions

aws partner network | saas factory

Breakouts

- ARC410 – Serverless SaaS deep dive: Building serverless SaaS on AWS
- ARC210 – Microservices decomposition for SaaS environments
- GPSTEC337 – Architecting multi-tenant PaaS offerings with Amazon EKS

Chalk talks

- ARC413 – SaaS metrics deep dive: A look inside multi-tenant analytics
- GPSTEC306 – Reinventing your technology product strategy with a SaaS delivery model
- ARC305 – Migrating single-tenant applications to multi-tenant SaaS
- API308 – Monolith to serverless SaaS: Migrating to multi-tenant architecture

Workshops

- SVS303 – Monolith to serverless SaaS: A hands-on service decomposition
- ARC308 – Hands-on SaaS: Constructing a multi-tenant solution on AWS

Builders sessions

- ARC405 – Building multi-tenant-aware SaaS microservices

Thank you!

Judah Bernstein
judahb@amazon.com

Ranjith Raman
ranraman@amazon.com

Please complete the session survey in the mobile app.

AWS re:Invent

aws