IF YOU BUILD IT, THEY WILL COME - A PLATFORM MODERNIZATION JOURNEY

TOPICS

Challenges in the existing on-premise environment

Planning for Success

Building our own PaaS

Designing for operational efficiency and shifting left

Tooling selection and what problem it solved

CHALLENGES

USABILITY CHALLENGES

Poor performance: Operational inefficiencies due to outdated versions of Docker Swarm & Rancher.

Difficult Upgrades: Application updates often required downtime, leading to service interruptions.

Complex Management: Customer facing apps required significant manual effort for troubleshooting which led to slow incident response times.

Limited developer and app support team access and experience with container environments

OBSERVABILITY CHALLENGES

Metrics collection required manual effort

No “out of the box” configuration

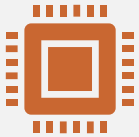
Lack of centralized logging and standard configurations

Poor visibility into the health of all containers

Production issues went unnoticed until they caused service disruptions.

TECHNICAL DEBT

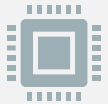
Technical debt due to old version of Docker Swarm and Rancher



Compatibility issues between legacy applications and newer environments resulted in frequent downtimes

Integrating modern tools with legacy systems limited ability to leverage newer technologies

INFRASTRUCTURE SCALABILITY



Difficult to scale due to manual effort.

Outage impact too large due to shared compute resources in cluster

Manual interventions using daemonsets to shift workloads

Inefficient scaling led to poor customer experience

PLANNING FOR SUCCESS

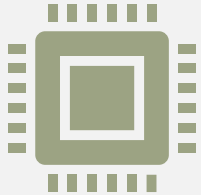
BUILD OUR OWN PAAS

Streamlined application deployment

A PaaS solution provides a consistent platform for application deployment across multiple environments, ensuring easier transitions from development to production

Increased development productivity

With automated deployment pipelines, integrated monitoring, and built-in security, developers can release updates more rapidly and confidently.



Abstracts infrastructure management

PaaS simplifies application deployment and maintenance by abstracting much of the underlying infrastructure, allowing developers to focus more on coding.

IMPORTANCE OF CUSTOMER COLLABORATION

Collaborate with your users/customers during all stages of build and run!!!

Feedback in planning stage provides view of customer needs, plans, and potential future enhancements

Let them in while you build your development environment

Keep collecting feedback as you build and implement new functionality

Higher end user satisfaction

USABILITY ENHANCEMENTS

Early Identification of "White Glove" service requirement

Build a training model

Increase developer velocity by removing barriers (give devs more permissions)

Decrease developer dependency on platform team (give devs more permissions)

Create patterned deployment models to keep control and clarity for supported deployment patterns

CLUSTER AND PLATFORM ADMINISTRATION ENHANCEMENTS

Reduce operational overhead = higher efficiency

Use managed services (EKS, RDS, etc)

Environmental isolation (Compute, Network, etc)

Reduce the blast radius

Automate your infrastructure and configuration deployments

Repeatable processes = patterns and less config drift

Velero for back up and restores current cluster configurations

OBSERVABILITY IMPROVEMENTS

Tagging all resources and importing data into Cloudability

Pay per use model passed down to “tenants”

Observability “out of the box” for developers

In Cluster OTEL collection of logs → Splunk

Tetrate UI for network traffic analysis in the Mesh

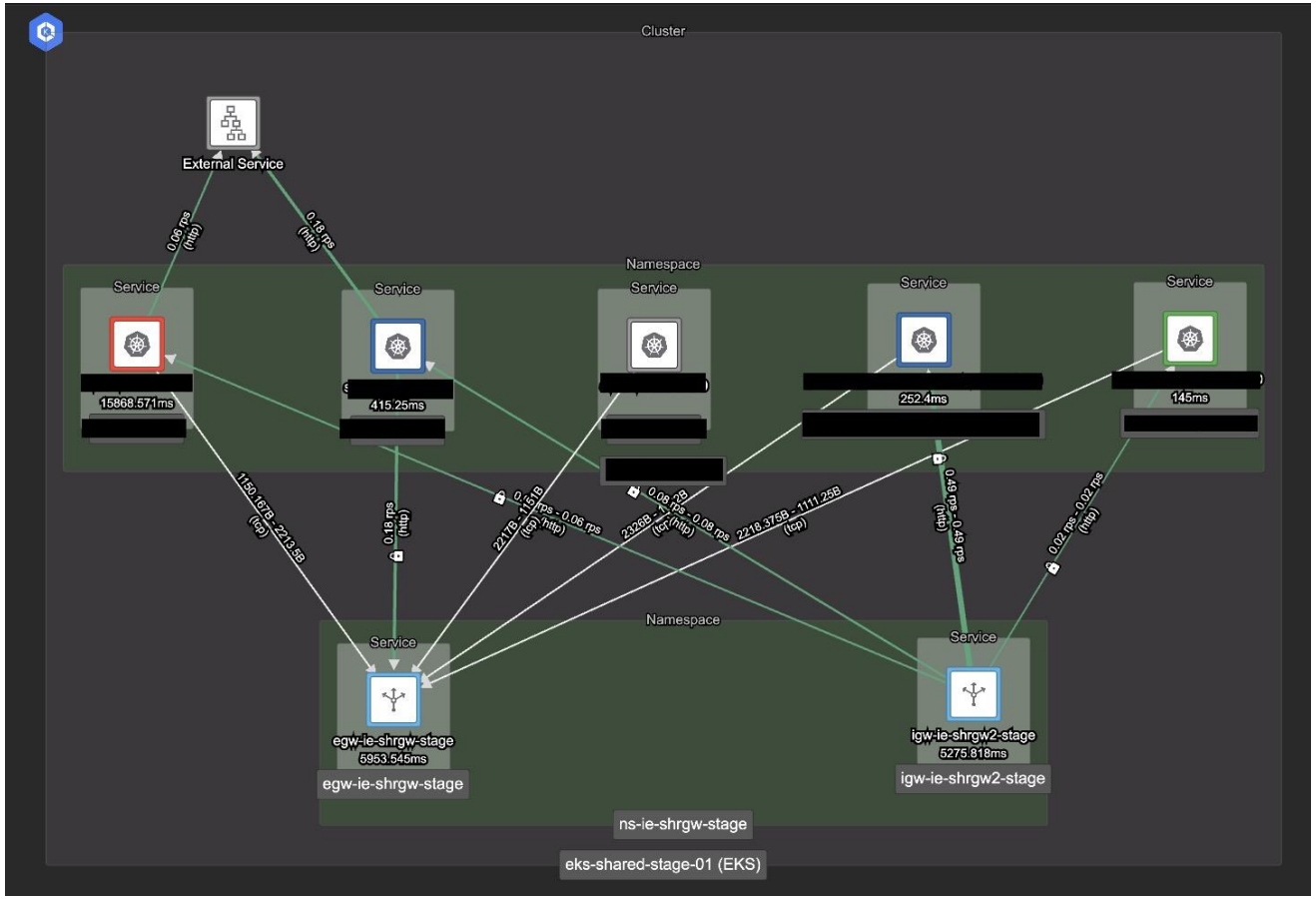
Easier Incident Management

Stakeholder Identification (tagging)

ZERO SETUP OBSERVABILITY

- Platform build includes automatic integration to Splunk and SIEM
  - Nodes
  - Pods
  - Gateways

- Default UI metrics available in Tetrate UI
  - Istio underneath
  - Traffic data pulled at the istio-proxy sidecar level for each services



SECURITY ENHANCEMENTS

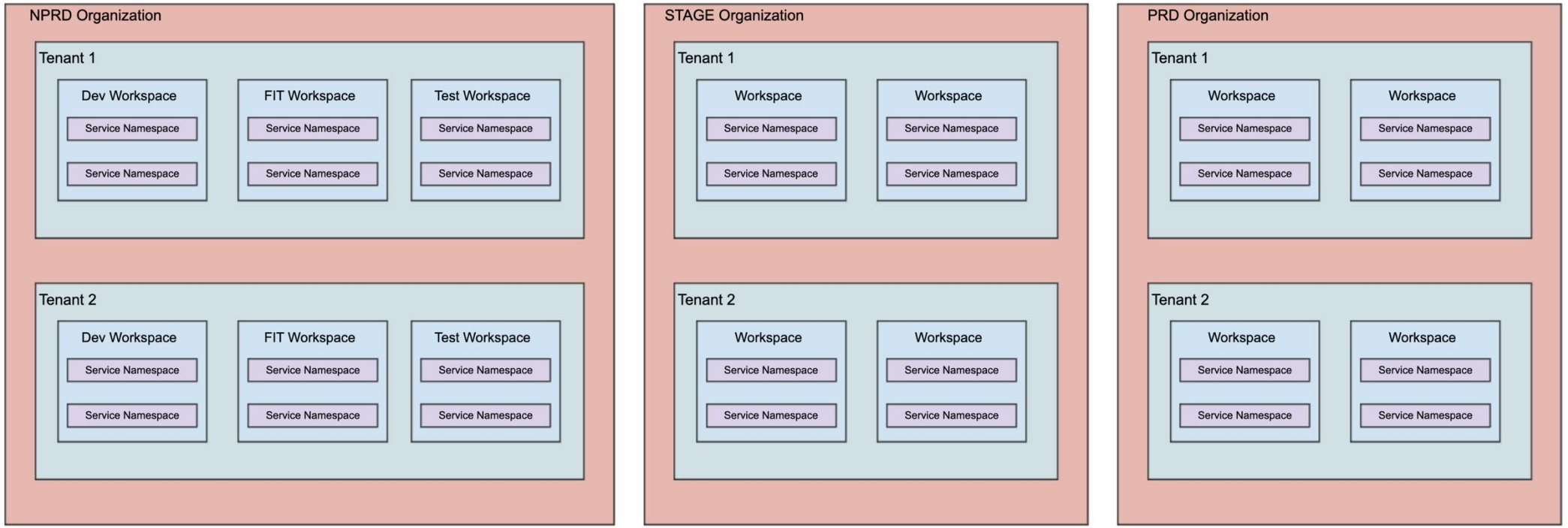
Compute and network isolation with Karpenter customizations

Careful permissions design for all platform components

- AWS, EKS, Istio, Tetrate

Twistlock provides continuous scanning real-time alerts for any potential threats in clusters

TENANCY MODEL



IF THIS IS PLATFORM AS A SERVICE…. WHAT ABOUT CLUSTER PERMISSIONS?

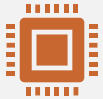
Tenant Privileged Account Group (CyberArk)	Tenant GitLab Deployment Agent	Tenant Domain Accounts Group (NPRD)	Tenant Domain Accounts Group (Stage/UAT, PRD)	Name	API Version	NameSpaced	Kind
read	read	read	read	bindings	v1	TRUE	Binding
read	read	read	read	componentstatuses	v1	FALSE	ComponentStatus
partial	full	partial	read	configmaps	v1	TRUE	ConfigMap
partial	full	partial	read	endpoints	v1	TRUE	Endpoints
partial	full	partial	read	events	v1	TRUE	Event
read	read	read	read	limitranges	v1	TRUE	LimitRange
read	read	read	read	namespaces	v1	FALSE	Namespace
read	read	read	read	nodes	v1	FALSE	Node
partial	full	partial	read	persistentvolumeclaims	v1	TRUE	PersistentVolumeClaim
read	read	read	read	persistentvolumes	v1	FALSE	PersistentVolume
partial	full	partial	read	pods	v1	TRUE	Pod
partial	full	partial	read	podtemplates	v1	TRUE	PodTemplate
partial	full	partial	read	replicationcontrollers	v1	TRUE	ReplicationController
partial	full	partial	read	resourcequotas	v1	TRUE	ResourceQuota
partial	full	partial	read	secrets	v1	TRUE	Secret
partial	full	partial	read	serviceaccounts	v1	TRUE	ServiceAccount
partial	full	partial	read	services	v1	TRUE	Service
read	read	read	read	clusterrolebindings	rbac.authorization.k8s.io/v1	FALSE	ClusterRoleBinding
read	read	read	read	clusterroles	rbac.authorization.k8s.io/v1	FALSE	ClusterRole
read	read	read	read	rolebindings	rbac.authorization.k8s.io/v1	TRUE	RoleBinding
read	read	read	read	roles	rbac.authorization.k8s.io/v1	TRUE	Role

INFRASTRUCTURE AND AUTO SCALING

Leverage the cloud (Multi zone compute for Mgmt and Workload nodes)

Enforce use of multiple availability zones

Pod Scaling control with HPA (Horizontal Pod Scaler)



Automatic, horizontal node scaling and consolidation provided by Karpenter

Customized for each tenant's needs

Self healing infrastructure (Never below minimum state)

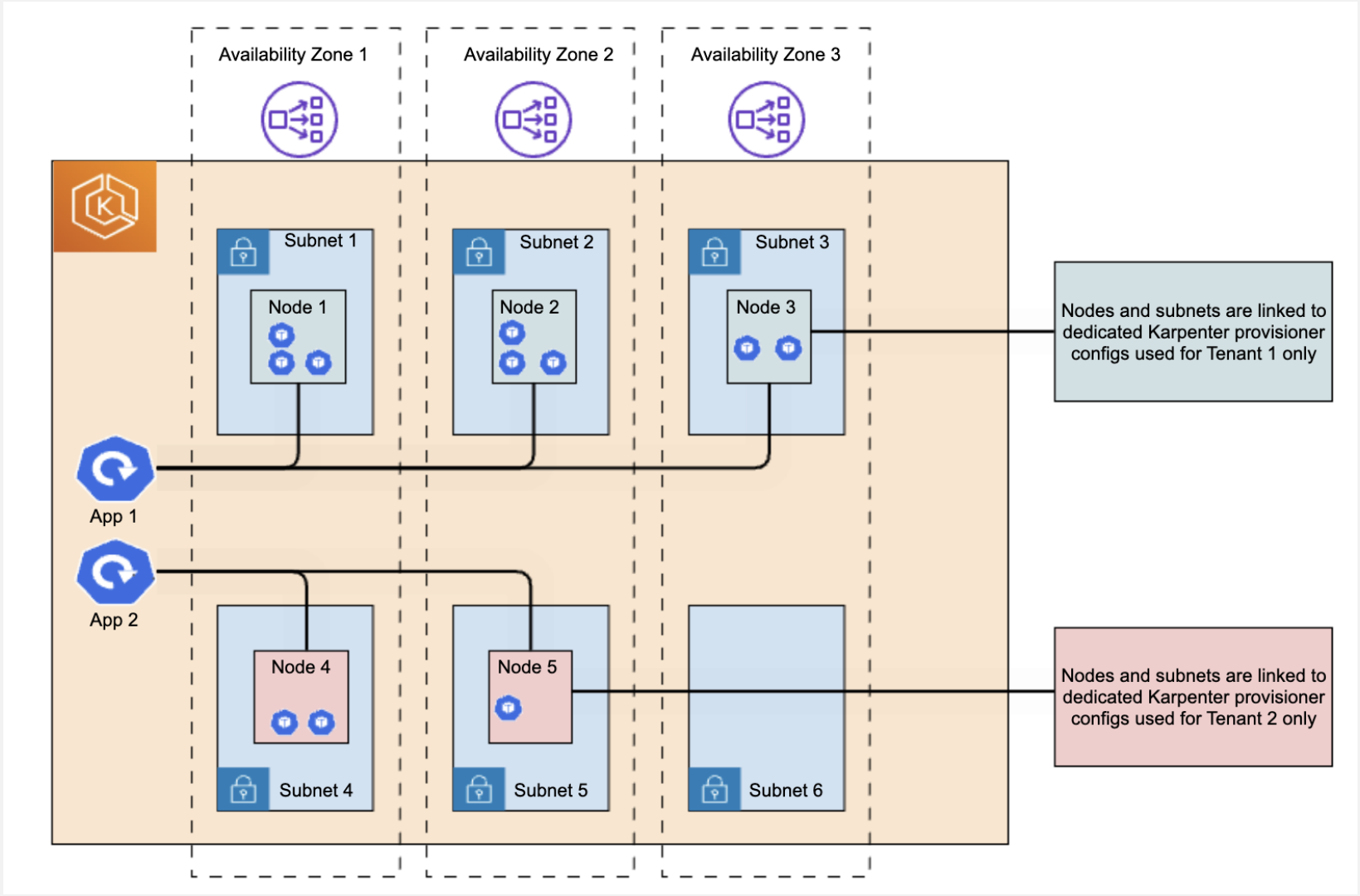
TELL ME ABOUT THIS NODE SCALING MAGIC

```yaml
---
apiVersion: karpenter.sh/v1beta1
kind: NodePool
metadata:
  name: "${cluster_name}-blue"
spec:
  template:
    metadata:
      labels:
        network: blue
    spec:
      nodeClassRef:
        apiVersion: karpenter.k8s.aws/v1beta1
        kind: EC2NodeClass
        name: "${cluster_name}-blue"
      taints:
      - key: network
        value: blue
        effect: NoSchedule

      requirements:
        - key: "karpenter.k8s.aws/instance-category"
          operator: In
          values: ["c", "m"]
        - key: "karpenter.k8s.aws/instance-cpu"
          operator: In
          values: ["8", "16", "32"]
        - key: "topology.kubernetes.io/zone"
          operator: In
          values: [${azs}]
        - key: "kubernetes.io/arch"
          operator: In
          values: ["amd64"]
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["on-demand"]

  disruption:
    consolidationPolicy: WhenUnderutilized
    expireAfter: 720h
  limits:
    cpu: "1024"
    memory: 2048Gi
```

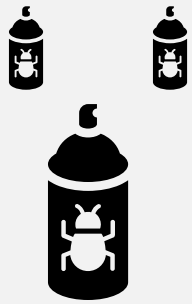
```yaml
---
apiVersion: karpenter.k8s.aws/v1beta1
kind: EC2NodeClass
metadata:
  name: "${cluster_name}-blue"
spec:
  role: "karpenter-myCluster-nodes-127001"
  amiFamily: Bottlerocket
  subnetSelectorTerms:
    - tags:
        Name: "Private-${cluster_name}-blue-*"
  securityGroupSelectorTerms:
    - tags:
        karpenter.sh/discovery/${cluster_name}: "${cluster_name}"
  blockDeviceMappings:
    # Root device
    - deviceName: /dev/xvda
      ebs:
        volumeSize: 40Gi
        volumeType: gp3
        encrypted: true
    # Data device: Container resources such as images and logs
    - deviceName: /dev/xvdb
      ebs:
        volumeSize: 300Gi
        volumeType: gp3
        encrypted: true
  tags:
    karpenter.sh/discovery/${cluster_name}: '${cluster_name}'
    Name: 'karpenter-${cluster_name}-blue'
    Solution-Name: "Bug Free Service"
    Billing-Code: "MoarDosh"
    Solution-ID: "127001"
    Expiry-Date: "2030-12-31"
    Environment: "NPRD"
```

```yaml
spec:
  containers:
  - nodeSelector:
      network: blue
    tolerations:
      - key: network
        operator: Equal
        value: blue
        effect: "NoSchedule"
    topologySpreadConstraints:
      - maxSkew: 1
        topologyKey: "topology.kubernetes.io/zone"
        whenUnsatisfiable: ScheduleAnyway
        labelSelector:
          matchLabels:
            app: {{ .Values.deployment.appName }}
```

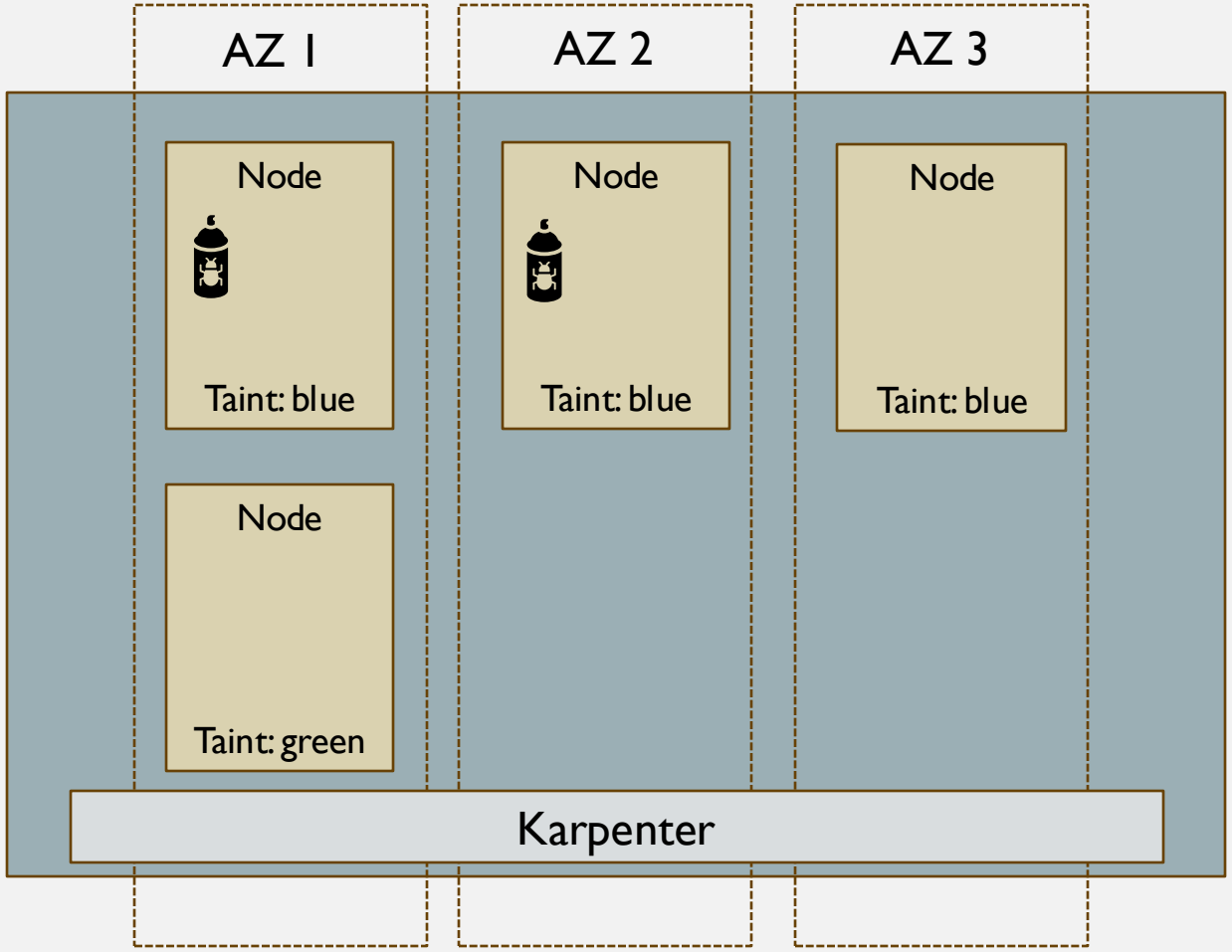
THAT'S COOL, WHAT ABOUT HA, COMPUTE ISOLATION, AUTO SCALING, COST EFFICIENCY



TELL ME MORE ABOUT THIS NODE SCALING MAGIC



Bug Free Service
tolerate: network: blue



AT LAST THE END!!!!!

- Hope you found this informative!!!!
- If you happen to run into us, we'd to chat more about our platform design!!!