Cloud Platforms

Dynatrace Training Module

Agenda

- Public Cloud Platforms
  - AWS
  - Google Cloud Platform
  - Microsoft Azure
- Enterprise Cloud Platforms
  - Cloud Foundry
  - Kubernetes
  - RedHat OpenShift
- Cloud Application Platforms
  - Heroku
- Virtualization
  - VMWare
- Misc
  - Docker
  - OpenStack

Public Cloud Platforms

AWS

AWS

- You can integrate Dynatrace with Amazon Web Services (AWS) for intelligent monitoring of services running in the Amazon Cloud
- AWS integration helps you stay on top of the dynamics of your data center in the cloud

- Before you can connect Dynatrace to AWS you need to configure your AWS account so that Dynatrace can access your CloudWatch metrics
- To do this you can either define a special role for Dynatrace or you can use a private access key
- Once your Amazon account is set, you need to tell Dynatrace how to access your Amazon data

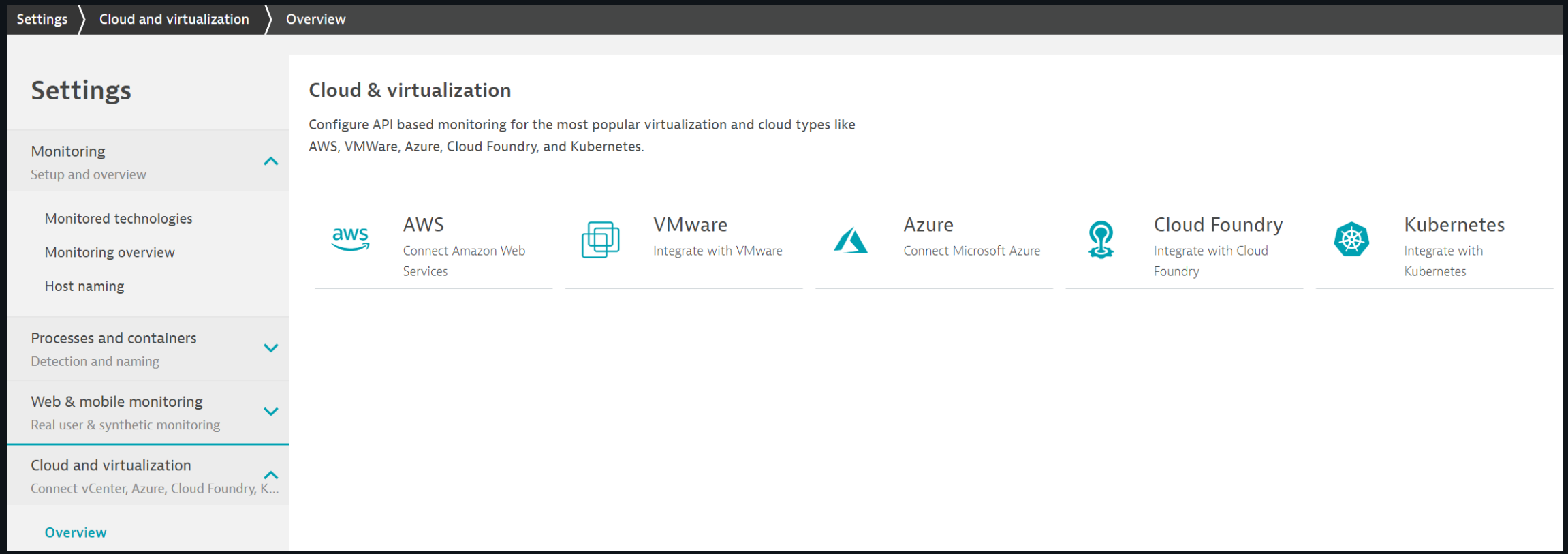
AWS

- First you need to enable access to your Amazon Account
- To get the information required for comprehensive AWS cloud-computing monitoring, Dynatrace needs to:
  - Identify all the virtualized infrastructure components that are in your AWS environment
  - Collect performance metrics related to those components
- We use this information to understand the context of your applications, services, and hosts

- For this to happen, you need to authorize Dynatrace to access your Amazon metrics
- You can enable Dynatrace access to your AWS metrics based on either user roles or access keys:
  - Create role-based access
  - Create key-based access

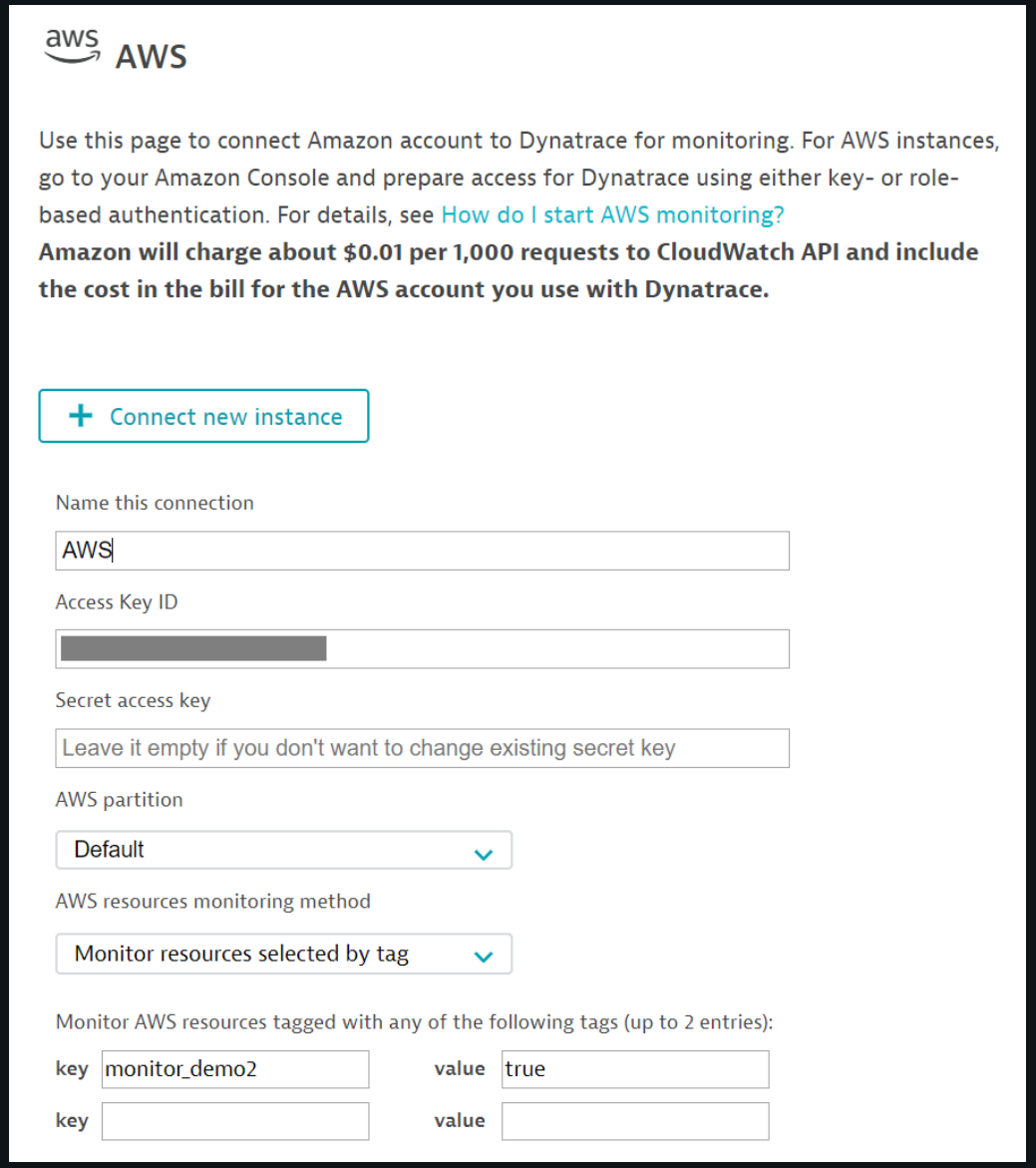
AWS

- Second you need to connect your Amazon account to Dynatrace
- This is done on the Settings → Cloud and Virtualization page



AWS

- Select a key-based or role-based connection
- Add AWS Tag Filters to limit API Calls

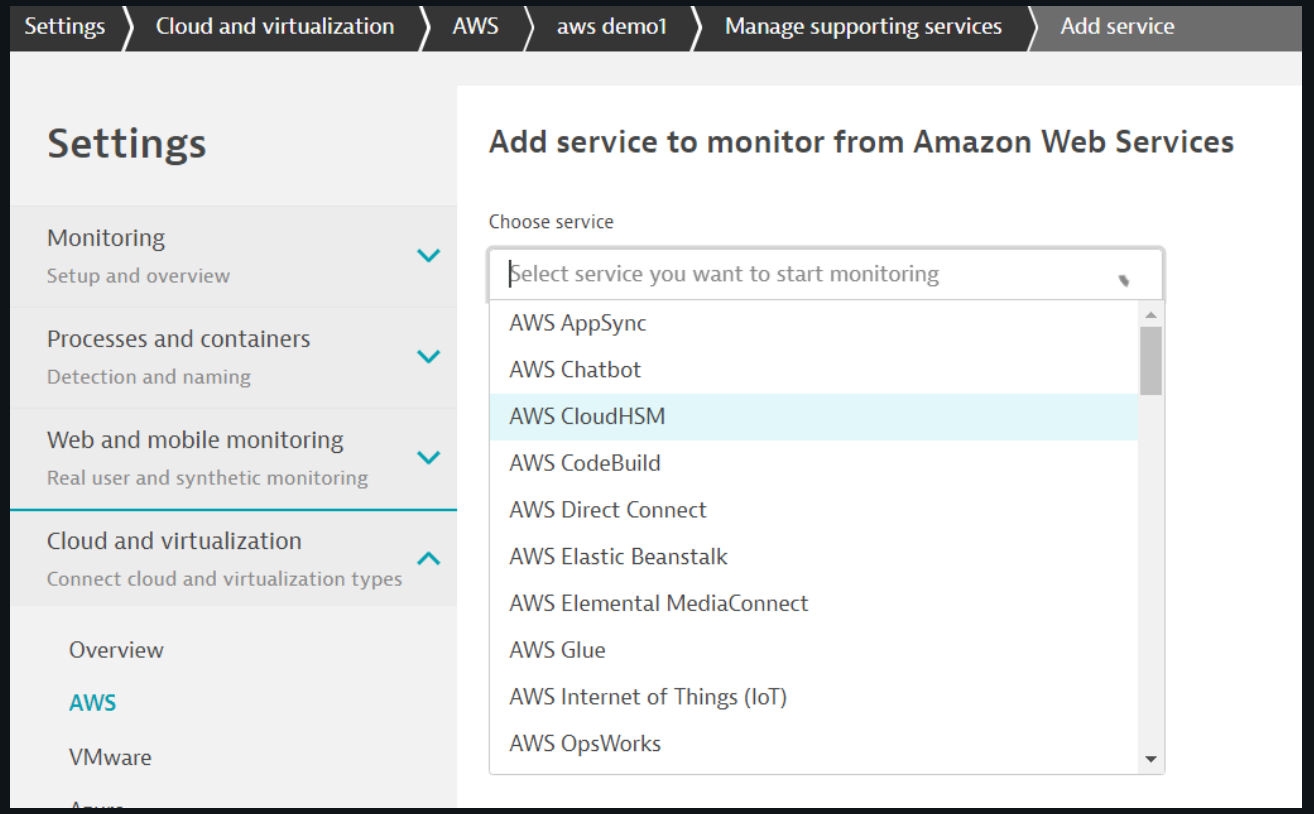


AWS Tagging

- By default, Dynatrace monitors all Amazon Web Services that have been specified in your permission policy
- Optionally, if you're interested in limiting API calls to Amazon, you can use tagging to limit the AWS resources that are monitored by Dynatrace

AWS Services

- Add monitoring for any AWS Services



Demo

Google Cloud Platform

GCP

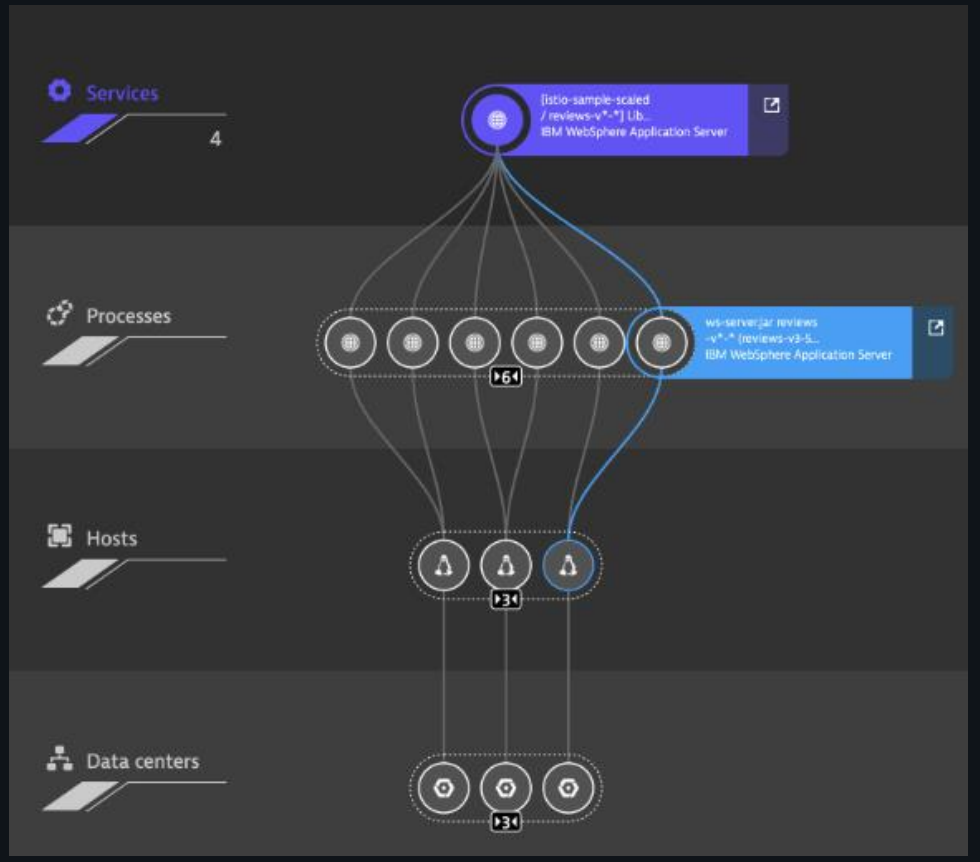
- Monitoring Google Compute Engine (GCE) instances works out of the box with the regular Linux and Windows OneAgent installers
  - OneAgent automatically reports GCE instance product information as well as important meta data such as:
    - GCE Zone
    - GCE Instance id
    - GCE Instance name
    - GCE Machine type
- Dynatrace supports monitoring of applications in Google App Engine flexible environment when deployed through custom Docker images
- Monitor Google Kubernetes Engine (GKE) clusters by rolling out OneAgent on each cluster node
  - Use the OneAgent Operator to automate the management, updates, and roll-outs of new OneAgent versions

GCE Instances

- All of this metadata can be used for
  - Tags
  - Naming rules
  - Management zones

GCP in SmartScape

- Google Compute instances and zones are first class citizens within the SmartScape



Demo

Microsoft Azure

Azure environments

Infrastructure-as-a-Serv

Virtual Machines
- Linux
- Windows

VM Scale Sets

Container-as-a-Service

Azure Container Service
- DC/OS and Marathon
- Docker Swarm
- Kubernetes

Platform-as-a-Service

Azure Cloud Services

App Services
- Web Apps
- API Apps
- Mobile Apps

Service Fabric

Functions (F-a-a-S)

Software-as-a-Service

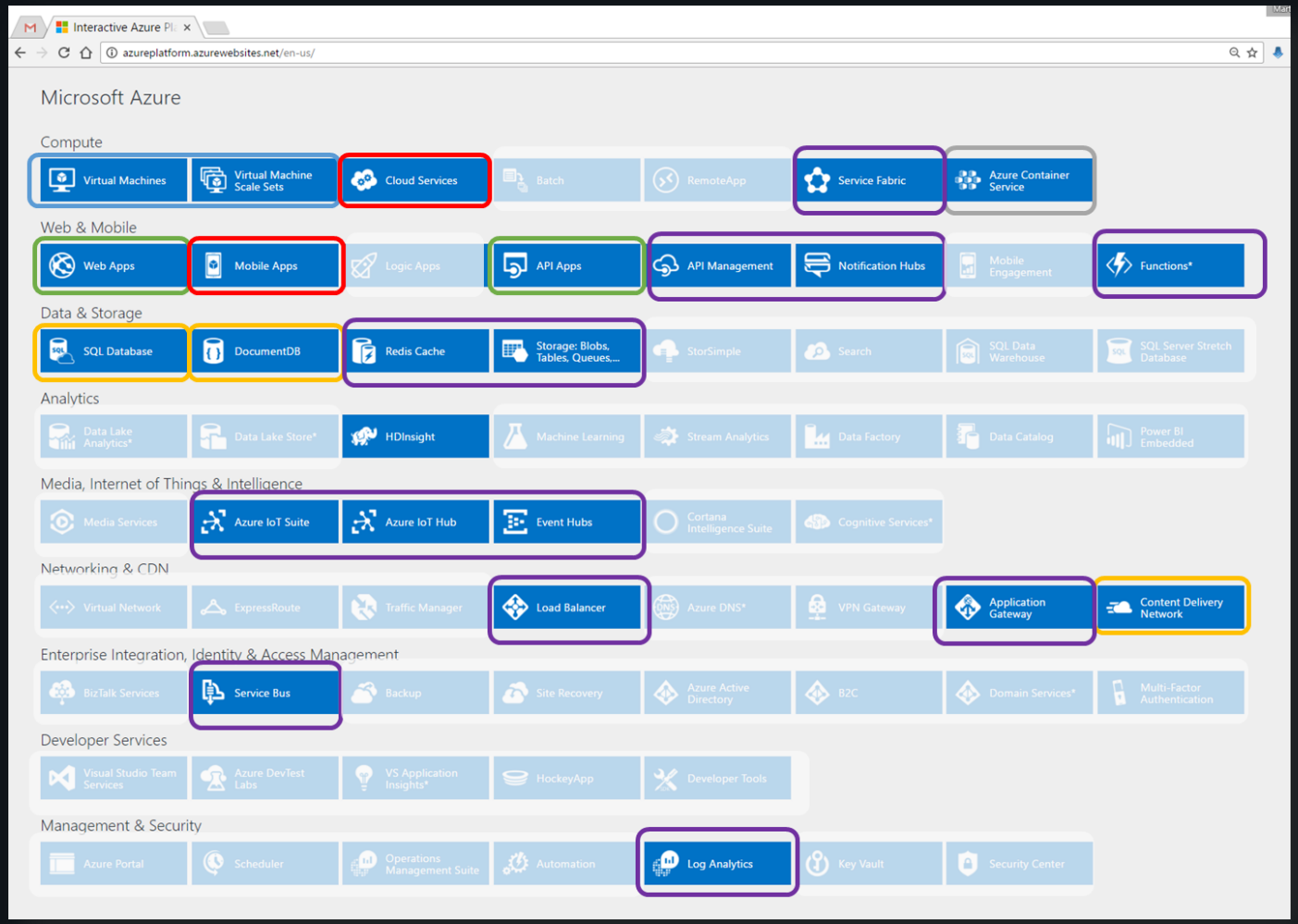
SQL-Server
DocumentDB
API Management
Redis Cache
Storage
Event Hubs
Service Bus

Active Directory

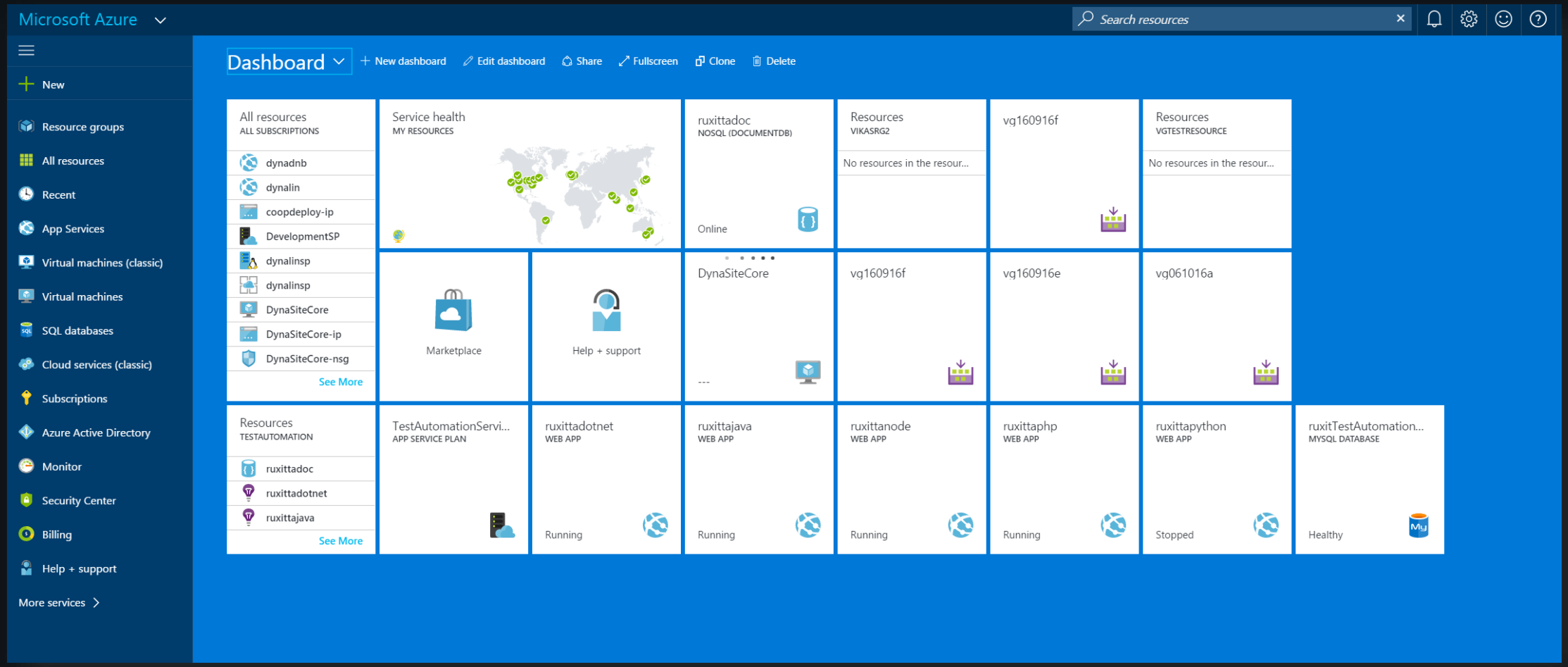
Get the full picture at http://azureplatform.azurewebsites.net/

What is Microsoft Azure

- Full-Stack Cloud Platform
  - on-prem: Azure Stack (ETA 2017)
  - IaaS, PaaS, FaaS, SaaS, Infra
- Windows, Linux, Containers

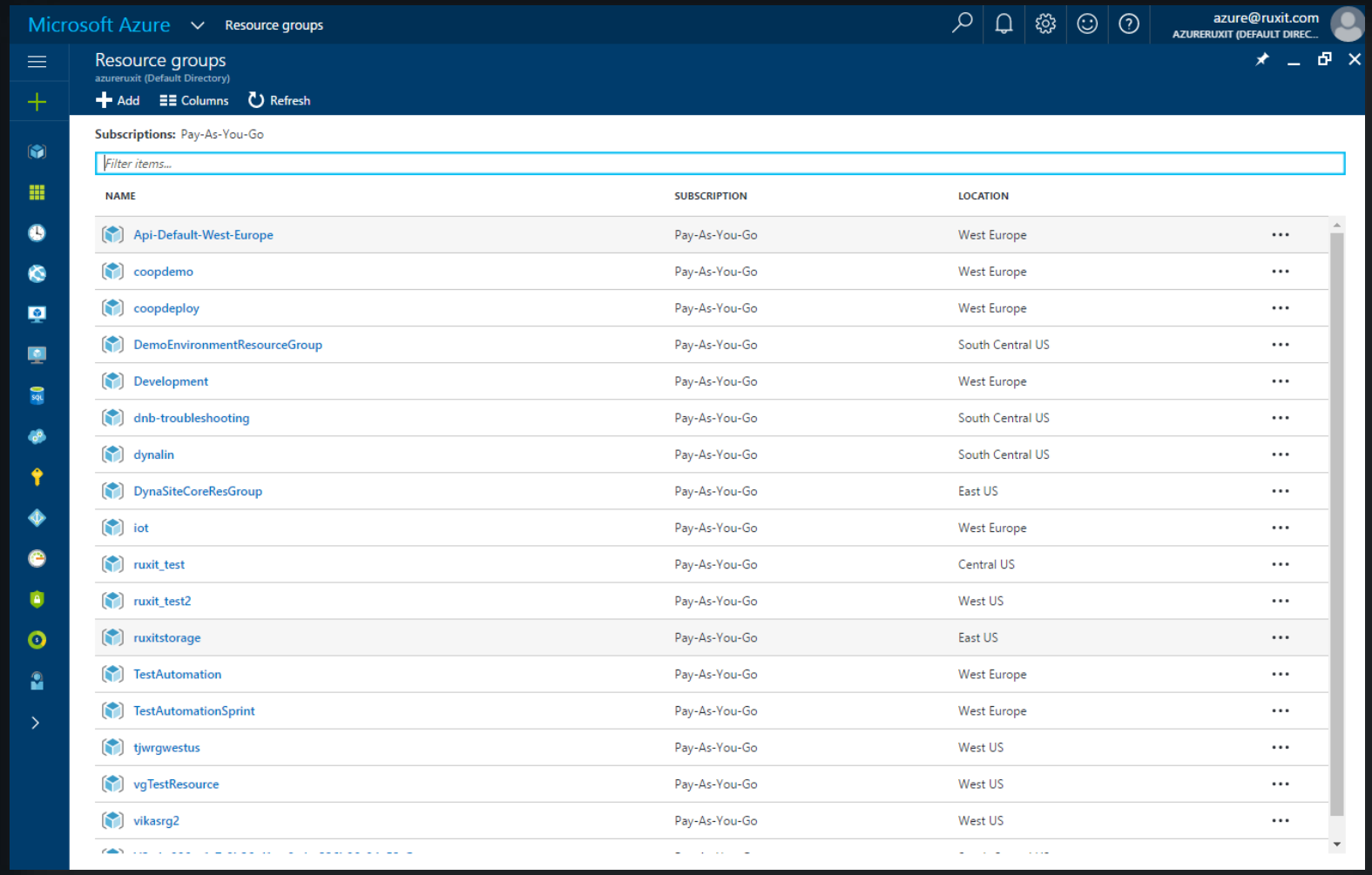


Azure Portal

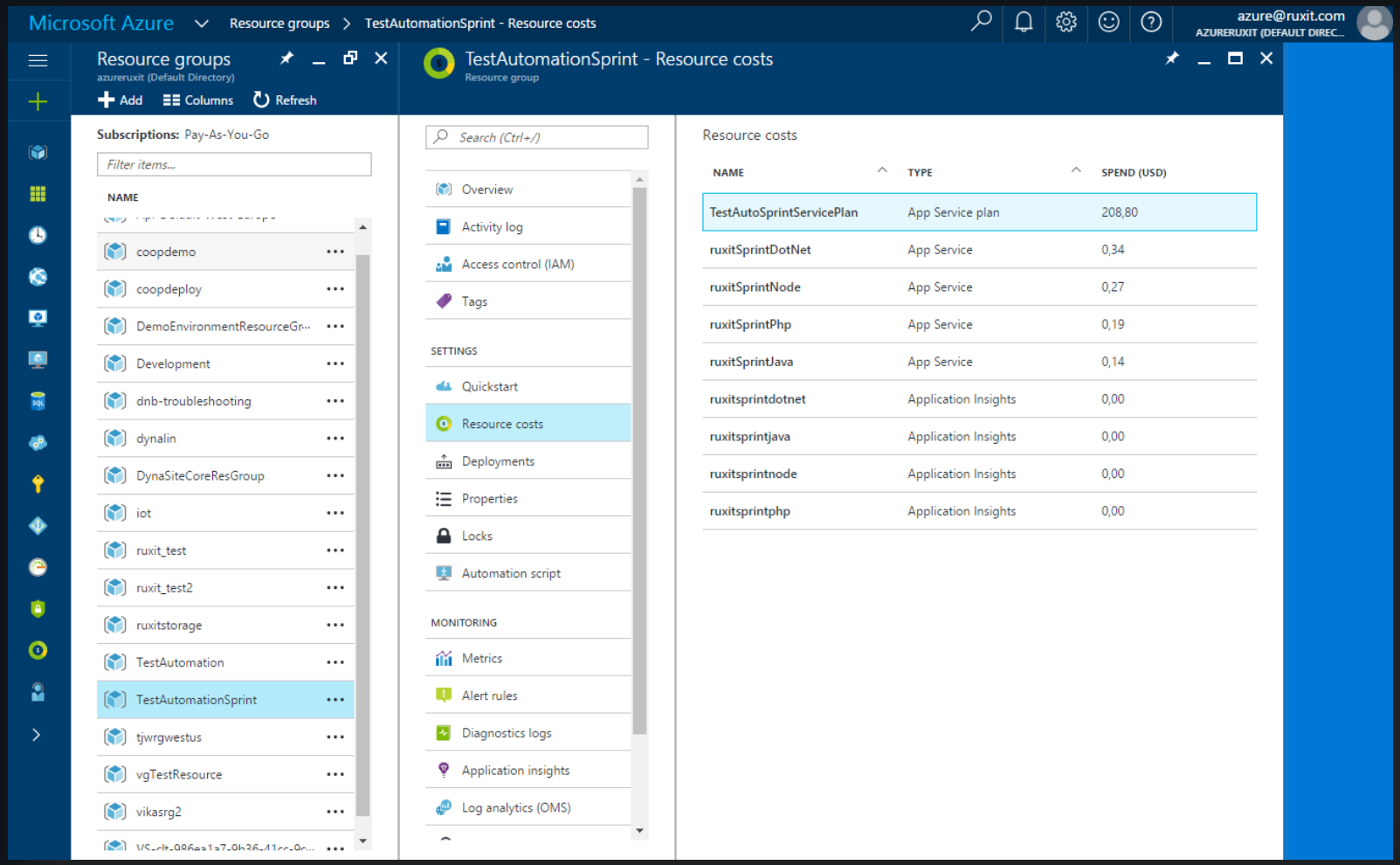


Resource Groups

- Primary objects in Azure
- Every Resource is assigned to a single ResourceGroup

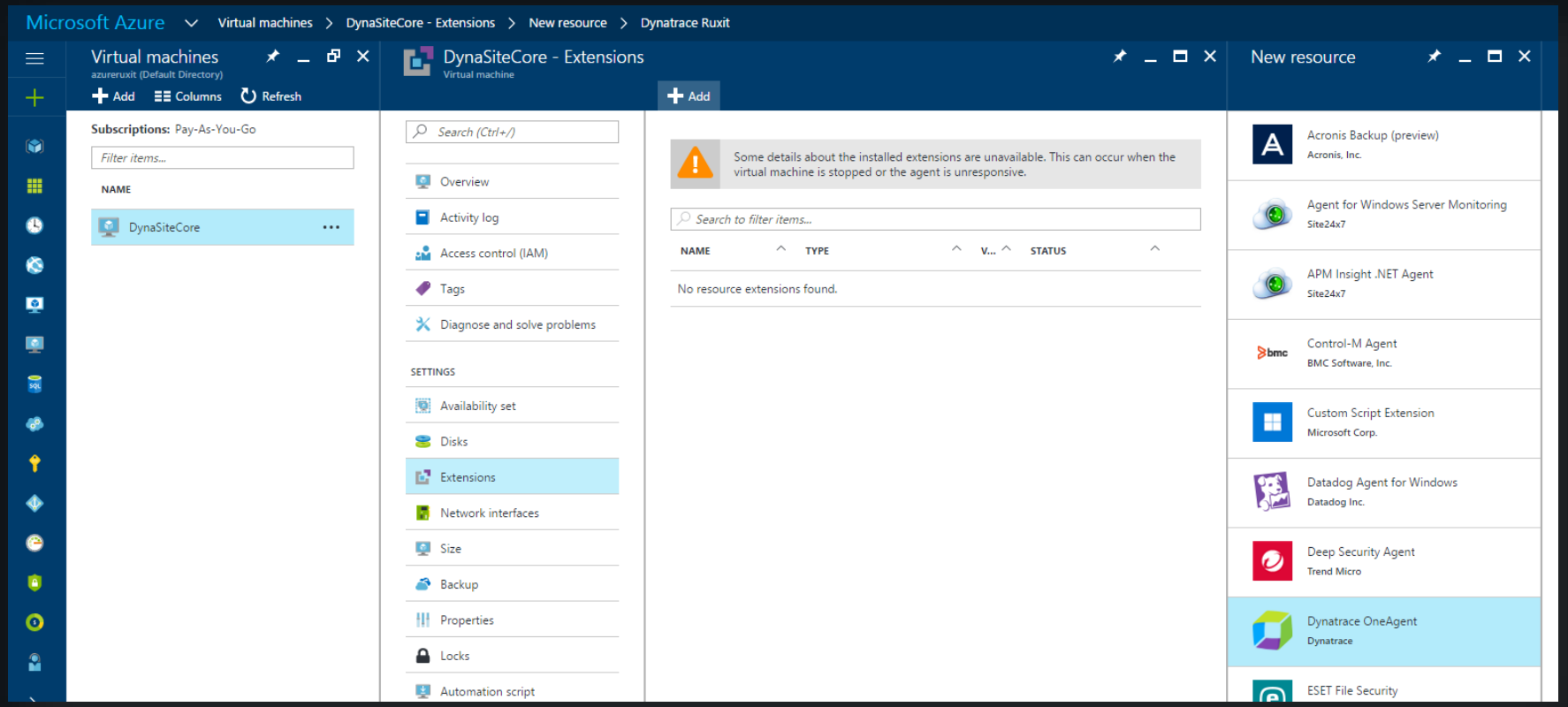


Resource costs based on Resource Groups



Extensions

- Allow for modifying existing resources on-the-fly without changing the base asset (eg VM, Web app)



Details about Azure

- Azure as it is now is the second generation already
  - the technology is called Azure Resource Manager (ARM)
  - first generation is now called Azure Classic (eg Classic VMs)
    - eg Azure Cloud Services is only available for Azure Classic → de facto deprecation
- Every Azure region runs at a defined revision
  - this is why certain features are not available in every region
    - App Insights will always run in NorAm regions, iirc
- Azure also provides Government clouds
  - instances separate from public cloud locations
- Contracts with governments assure data security
  - especially interesting for Europe-based companies

What and how do we monitor?

Compute service	Extensions and integration	Azure monitor metrics
Virtual Machines	VM-Extension[1]	yes
Virtual Machine Scale Set	VM-Extension[1]	yes
Service Fabric	VM-Extension[1]	
Azure Kubernetes Service (AKS)	VM-Extension[1]	
Azure Container Services/Kubernetes	VM-Extension[1]	
Azure Container Services/Docker Swarm	VM-Extension[1]	
Cloud-Services (Classic)	Startup-Script	
HDInsight	Startup-Script	
App Service (Windows based)	SiteExtension	Yes
Azure Functions	SiteExtension (BETA)	Yes

[1] Additional to the VM-Extension, standard OneAgent installation options are still available

Additional Metrics via Azure Monitor

Platform service	OneAgent code-module support	Integration of Dynatrace with Azure Monitor
Blob-Storage	HttpClient[1]	yes
Table-Storage	HttpClient[1]	yes
Queue-Storage	HttpClient[1]	yes
File-Storage	Infrastructure monitoring	yes
Disk-Storage	Infrastructure monitoring	yes
ServiceBus Queues and Topics	Microsoft Azure Service Bus Client for .NET	yes
Load-Balancer	Infrastructure monitoring	yes[3]
Application Gateway	Trace-Context[4]	yes
API Management	Trace-Context[4], SDK[5]	yes
SQL Azure	Supported database frameworks[2]	yes
CosmosDB	MongoDB API, Cassandra API, HttpClient[1]	coming soon

Additional Metrics via Azure Monitor

Platform service	OneAgent code-module support	Integration of Dynatrace with Azure Monitor
Azure DB for MySql	Supported database frameworks[2]	no
Azure DB for PostgreSQL	Supported database frameworks[2]	no
SQL Data Warehouse	Supported database frameworks[2]	no
SQL Server Stretch	Supported database frameworks[2]	no
Redis Cache	Supported client libraries	yes
Event Hubs	SDK[5]	coming soon
IoT Hub	Trace Context[4], SDK[5]	coming soon

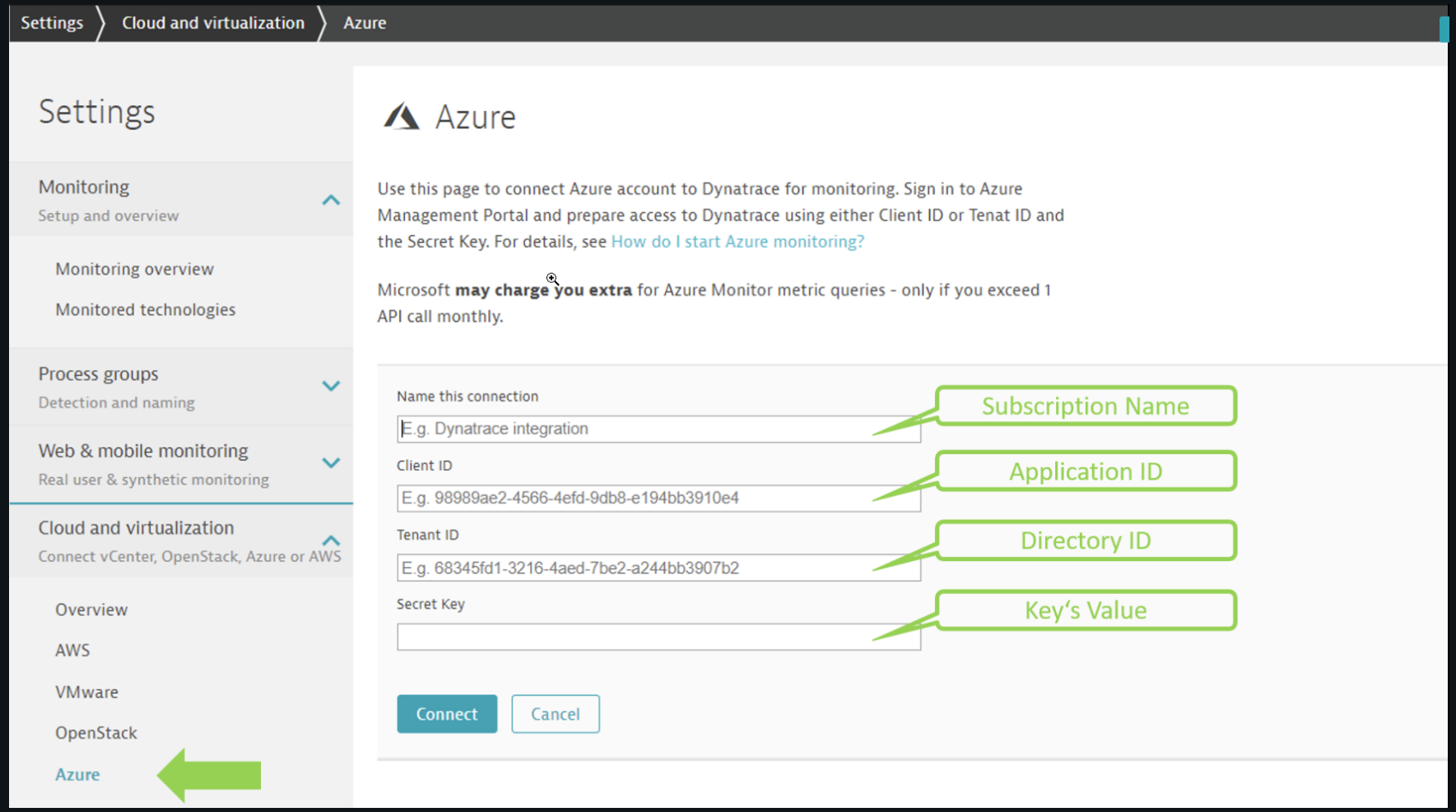
[1]Traces HTTP calls via HttpClient support
[2]Trace database calls via supported database frameworks (for example, ADO.NET or JDBC).
[3]Only available for Standard Load Balancer
[4]End-2-End tracing via Trace Context
[5]End-2-End tracing using OneAgent SDK

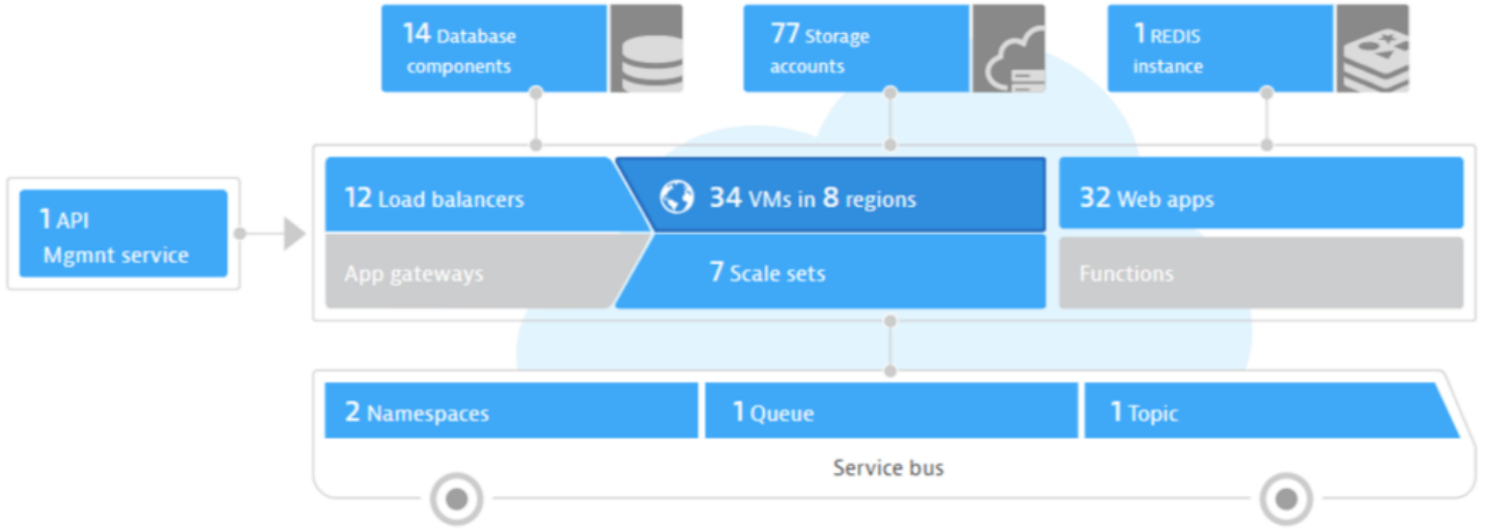
Deploying applications w/ Dynatrace

Deploying Monitoring for Azure

- Three aspects to monitor:
  - Enhanced Service Insights
  - Azure Service Fabric applications
  - Azure Web Apps
  - Azure Compute VMs

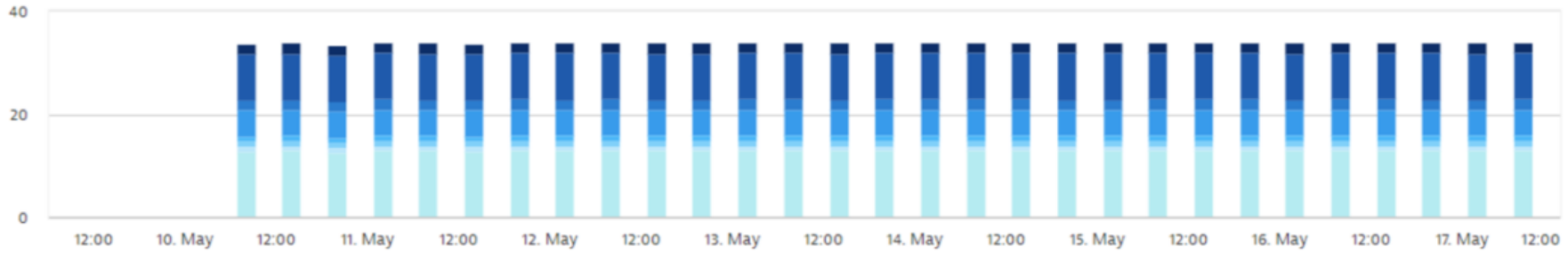
Enhanced Azure Service Insights





Environment dynamics

Average number of virtual machines over last 7 days



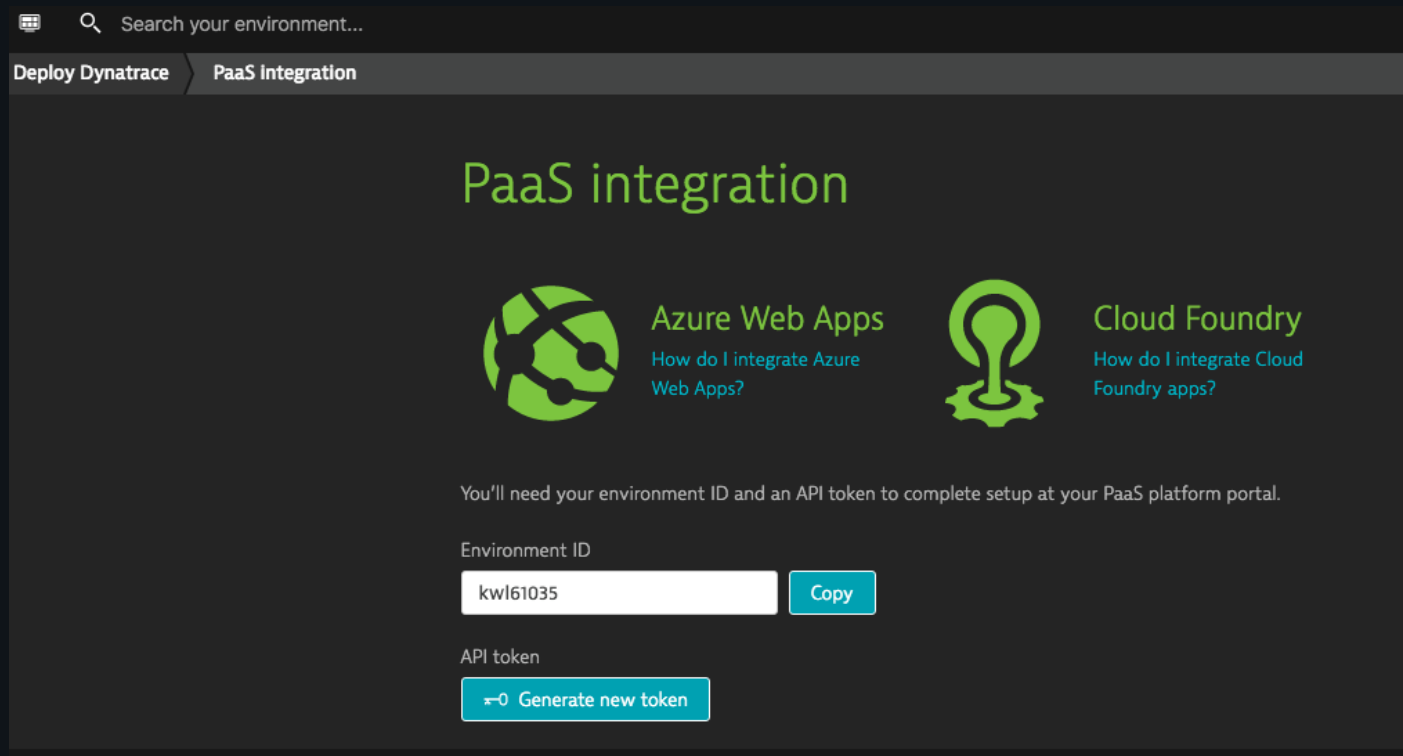
Virtual machines
+34
Compared to Wed, May 9

Region ▲	May 9	Change	Now
centralus	0	5	5
eastus	0	9	9
eastus2	0	1	1

Monitor Azure Service Fabric

Monitor the Service Fabric

- Step 1
  - Generate PaaS Token



Monitor the Service Fabric

- Install the VMSS OneAgent extension using Azure-Cli or PowerShell

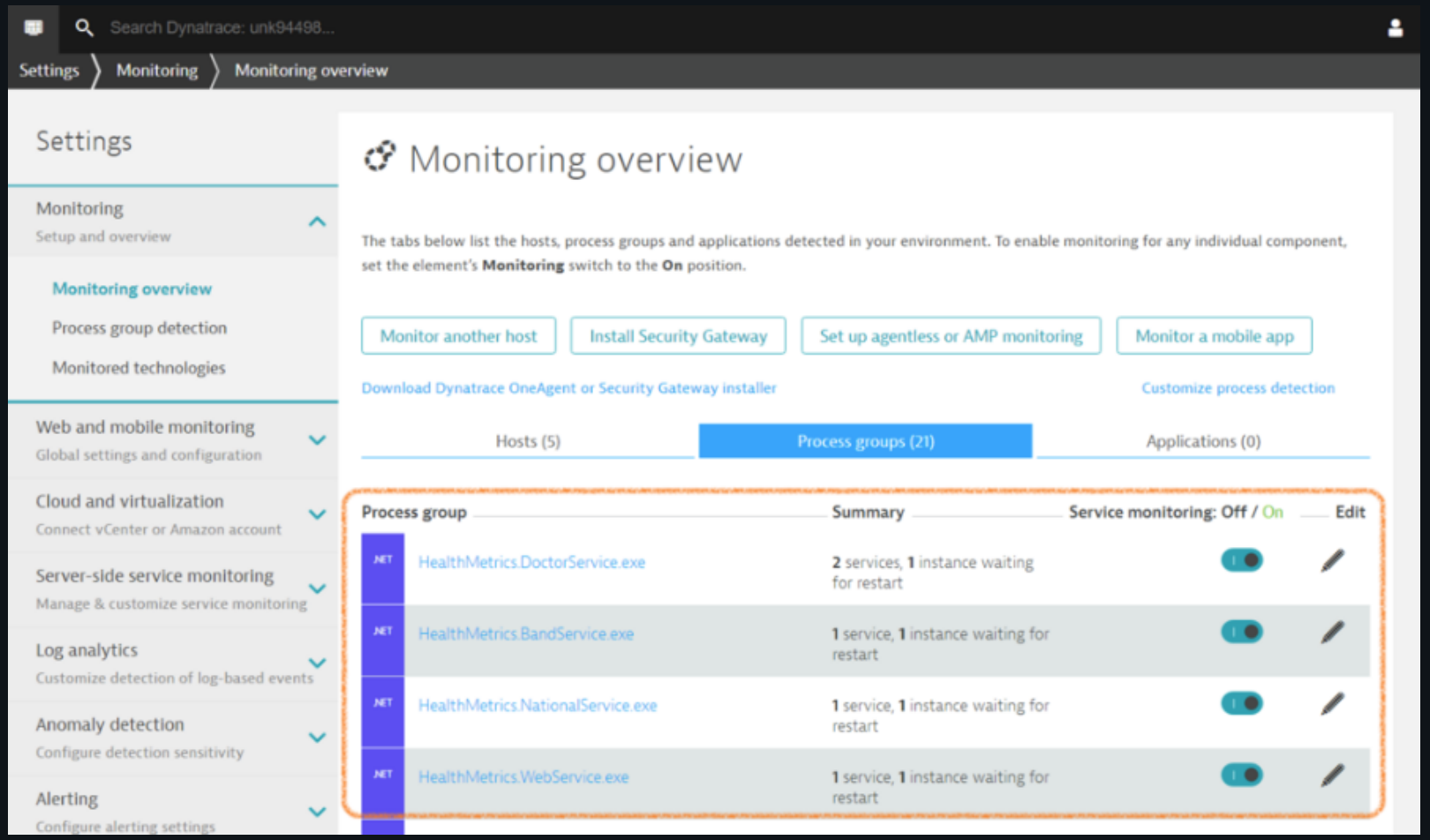
```
az vmss extension set
  -n oneAgentWindows
  --publisher dynatrace.ruxit
  -g "<resource group name>"
  --vmss-name "<VMSS name>"
  --settings "{\"tenantId\":\"<your environment ID>\",\"token\":\"<PaaS token>\"}"
```

Add the Dynatrace OneAgent Azure VM extension to the VMSS definition.

```
Add-AzureRmVmssExtension
  -Name oneAgentWindows
  -Publisher dynatrace.ruxit
  -Type oneAgentWindows
  -TypeHandlerVersion 1.99
  -AutoUpgradeMinorVersion $true
  -Setting @{ "tenantId"="<your environment ID>"; "token"="<PaaS token>" }
  -VirtualMachineScaleSet $vmss
```

Monitor the Service Fabric

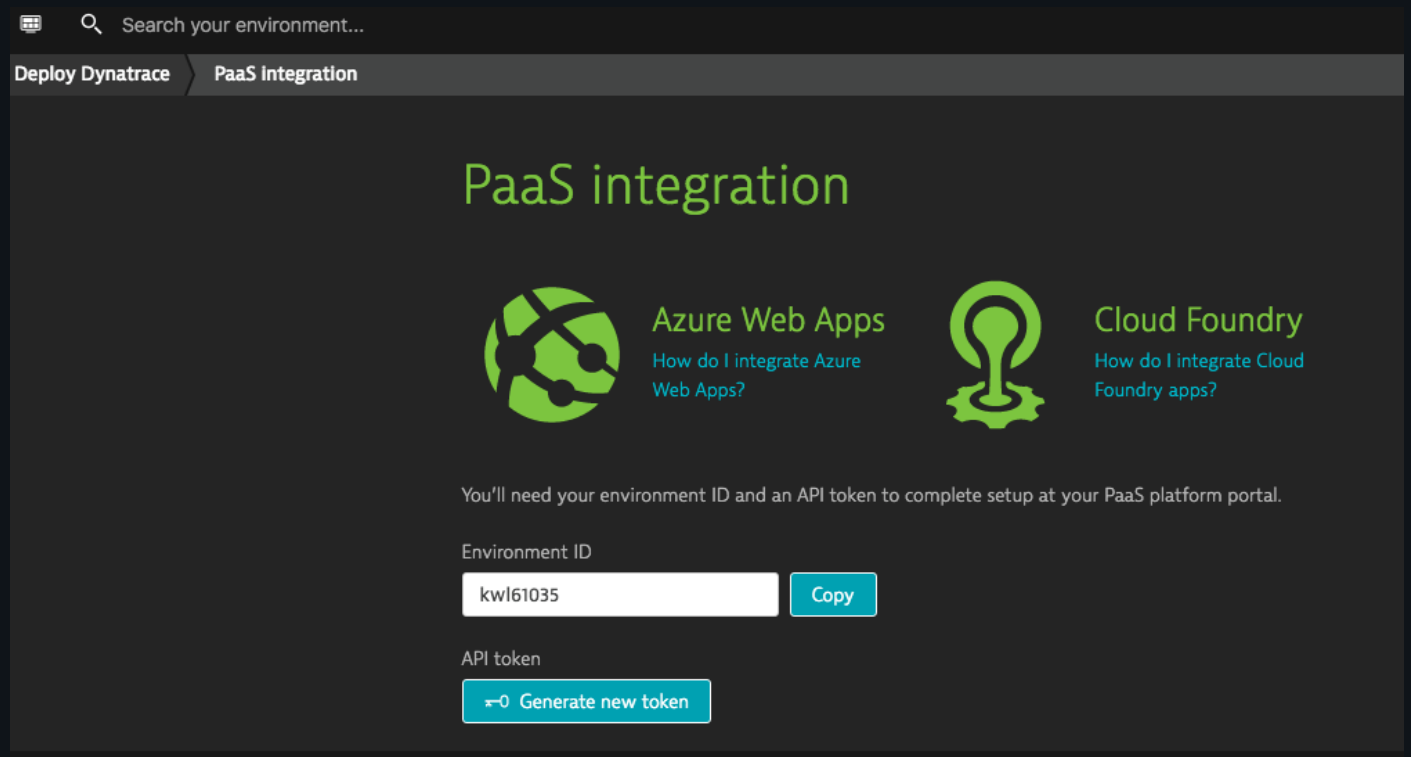
- Enable service monitoring for code-level visibility



Monitor Azure Web Apps

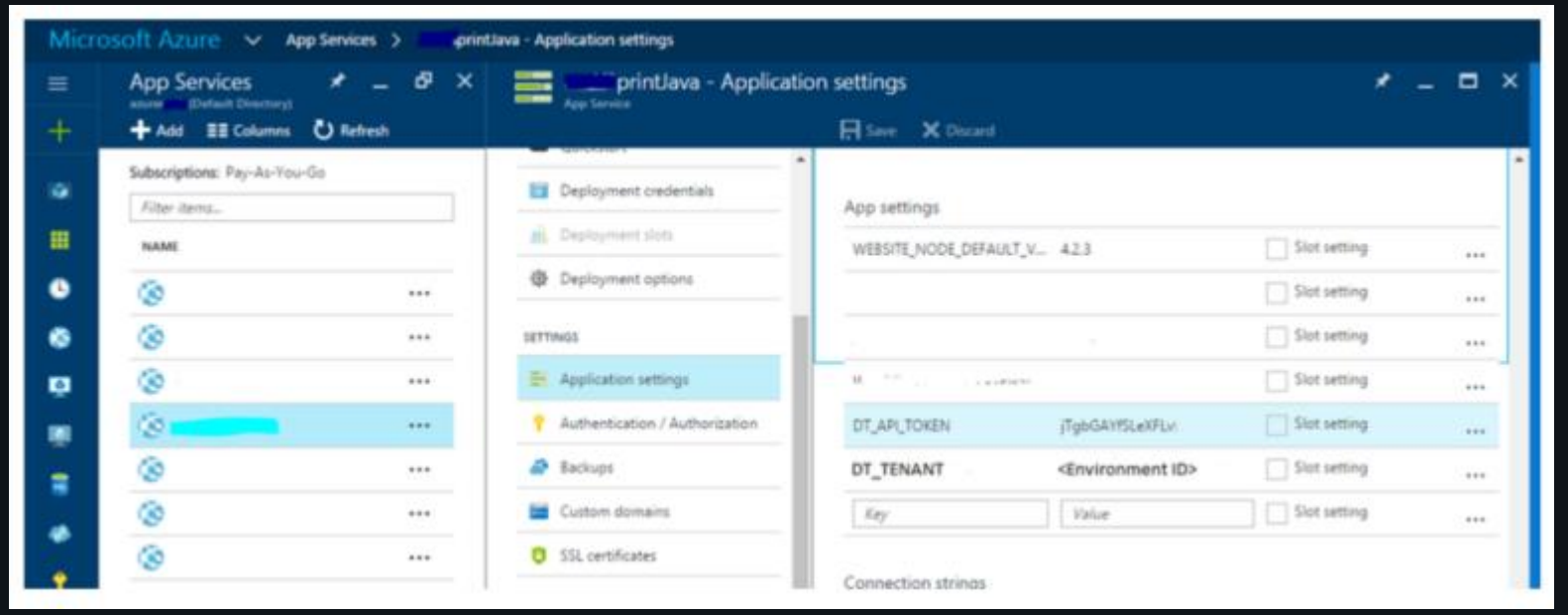
Monitor a Web App

- Step 1
  - Generate PaaS Token



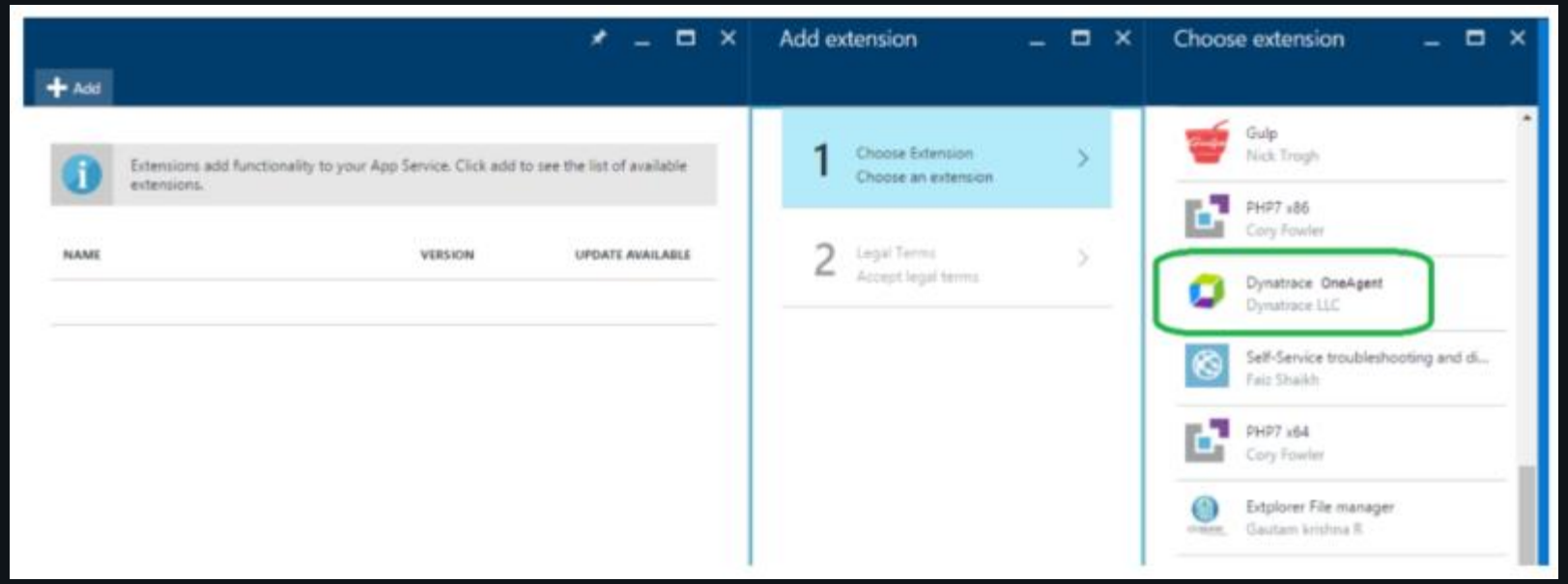
Monitor a Web App

- Step 2
  - Configure the Dynatrace Site Extension via the Azure portal



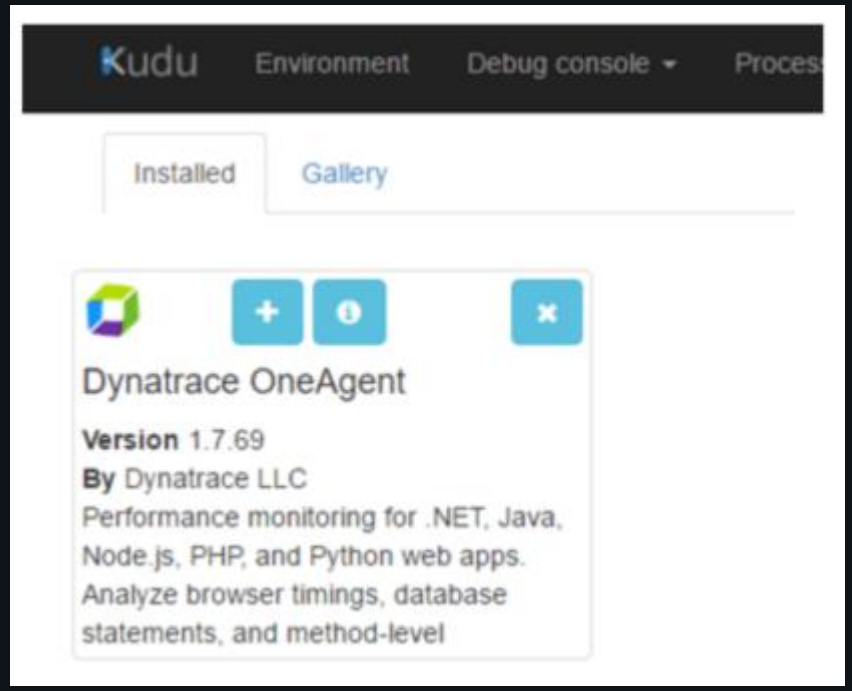
Monitor a Web App

- Step 3
  - Install the Dynatrace Azure site extension
  - Via the Azure Portal



Monitor a Web App

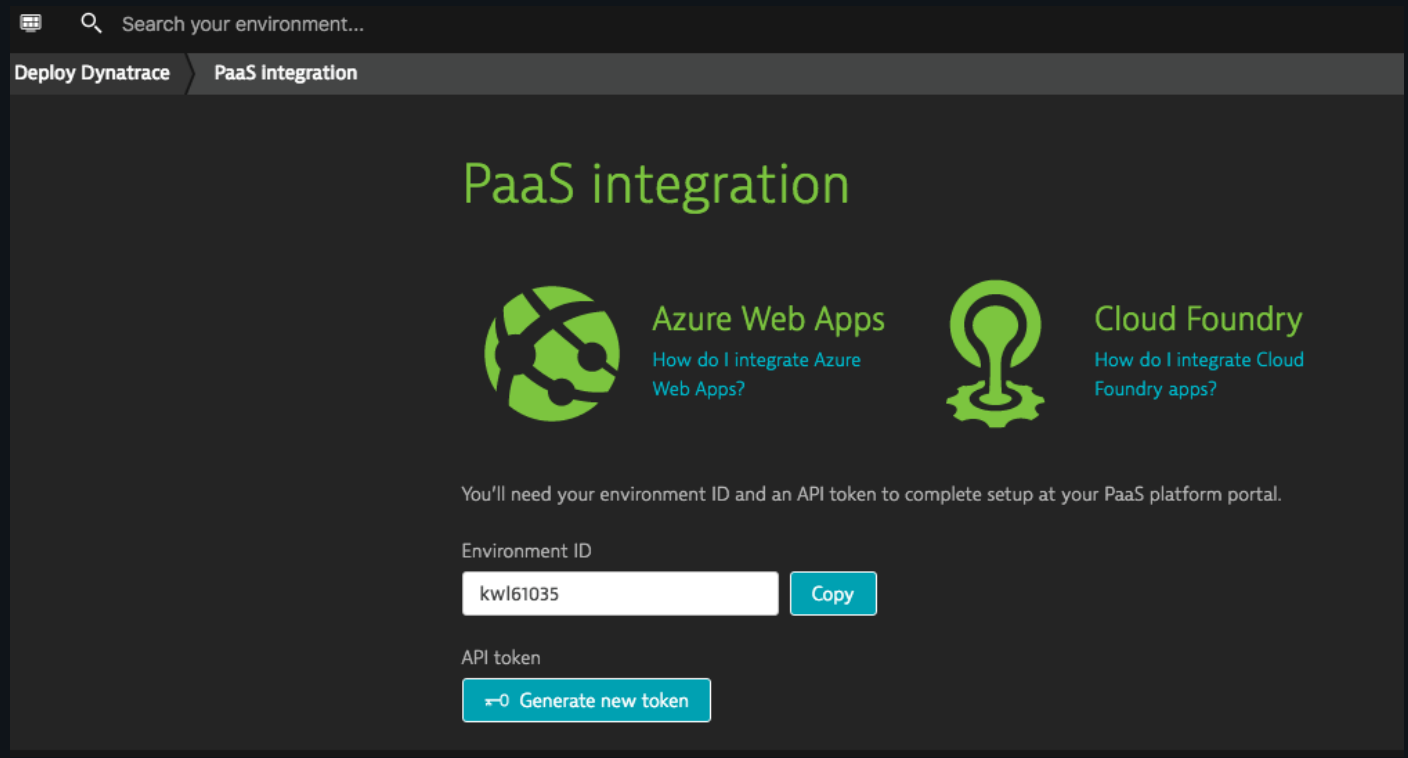
- Step 3
  - Install the Dynatrace Azure site extension
  - Via Kudo



Monitor Azure Compute VMs

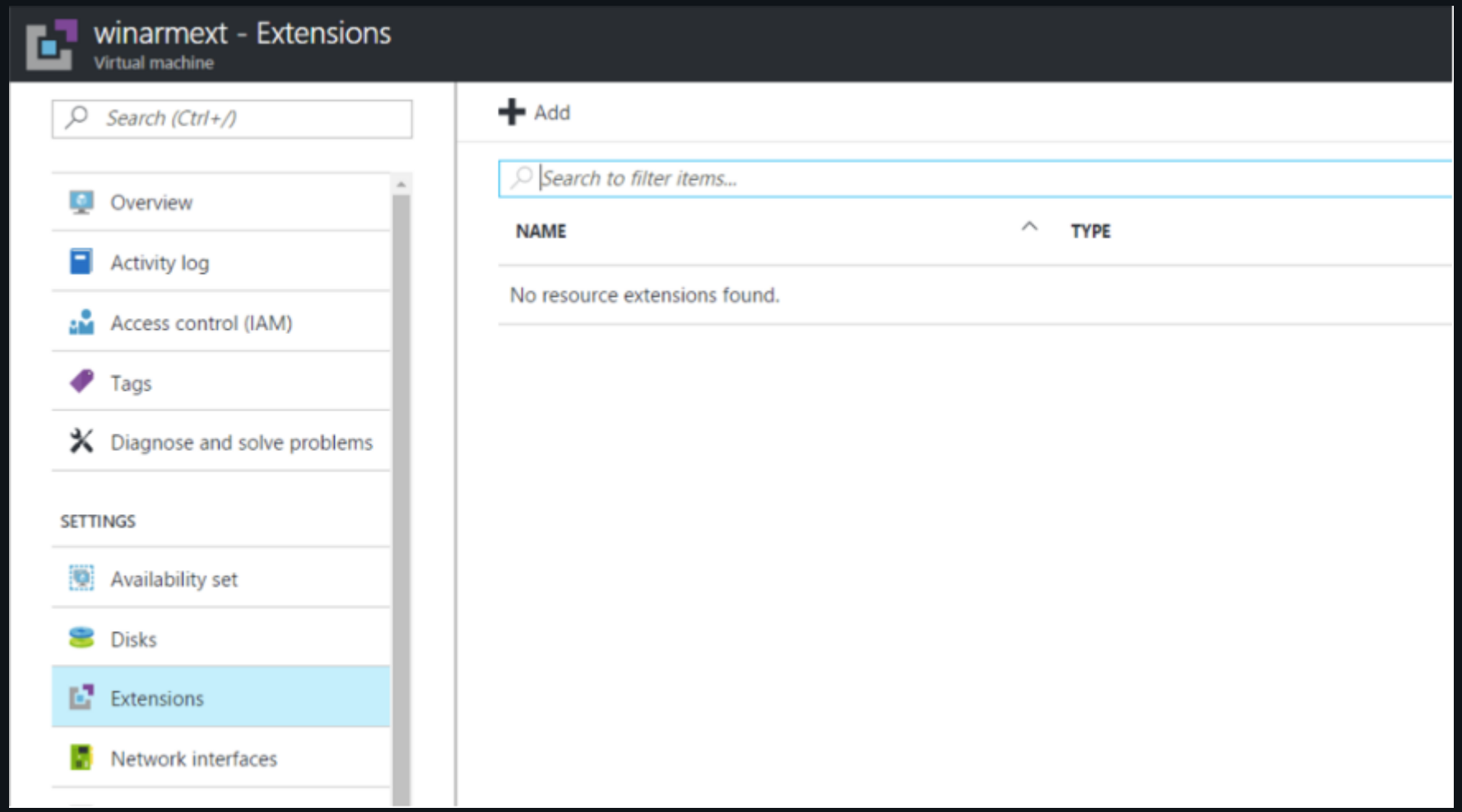
Monitor Compute VMs

- Step 1
  - Generate PaaS Token



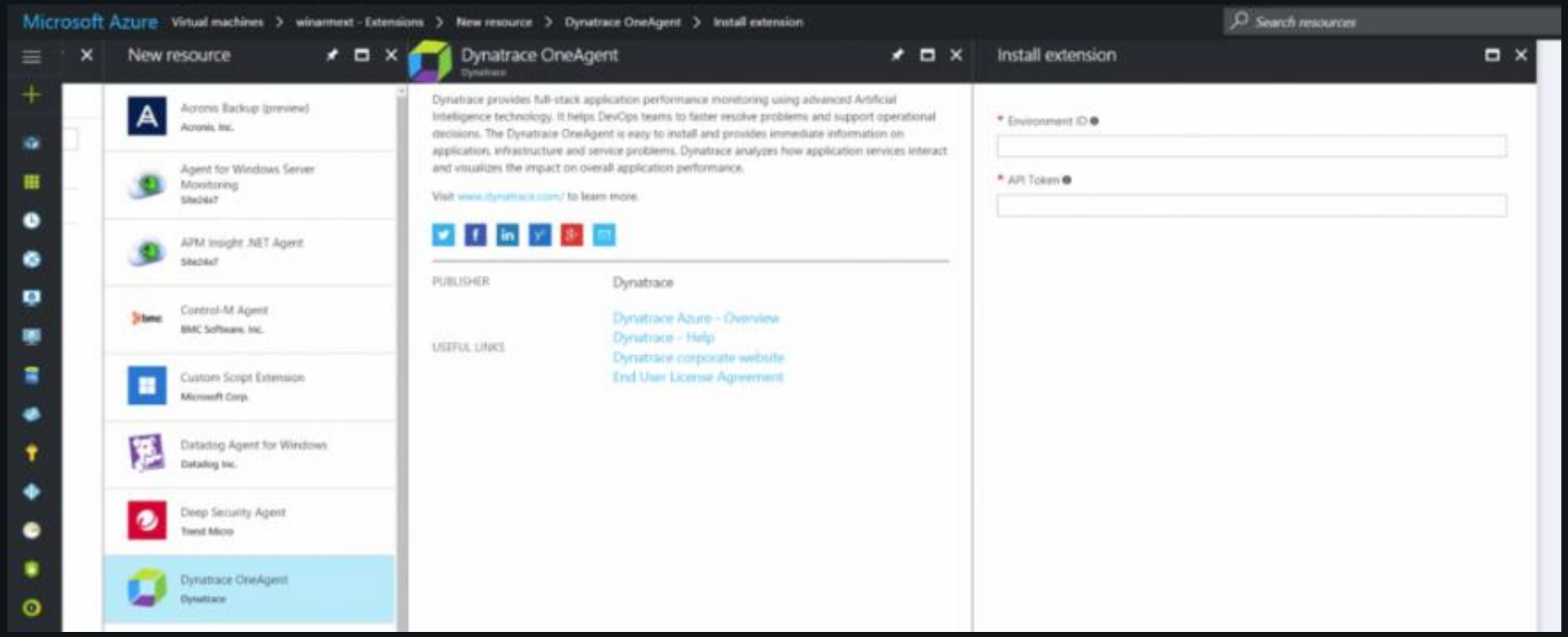
Monitor Compute VMs

- Install the Dynatrace OneAgent Azure VM Extension via Azure Portal



Monitor Compute VMs

- Configure the OneAgent Extension



Demo

Enterprise Cloud Platforms

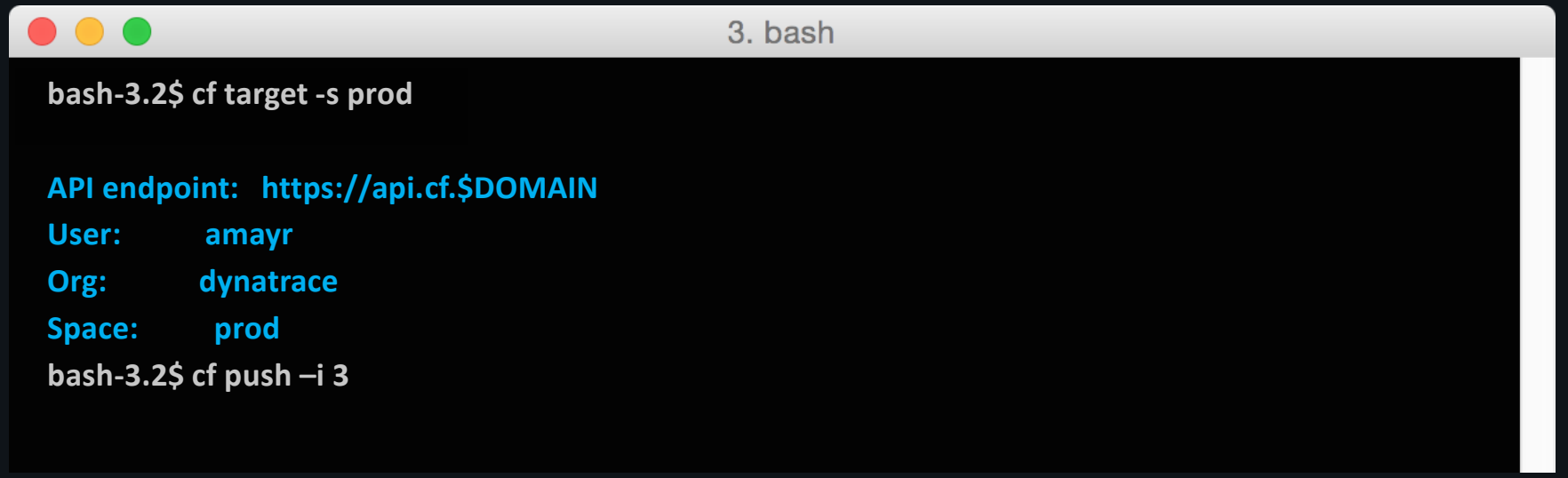
Cloud Foundry

What is Cloud Foundry?

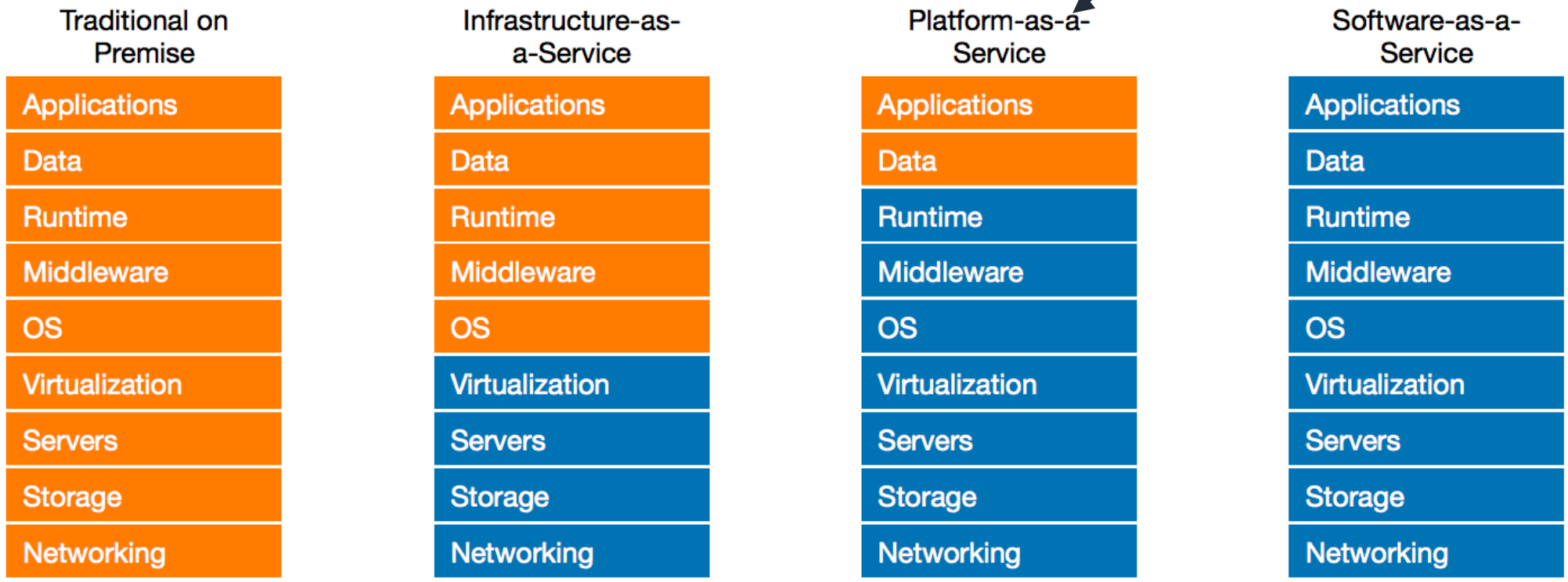
- An open source cloud **Platform as a Service**
- Enables developers to build, deploy, run, and scale apps easily
- Developers only care about their apps rather than underlying infrastructure and OS

"Here is my source code, Run it on the cloud for me, I do not care how" – Onsi Fakhouri

Scheduling, scaling, failover, health-management, load-balancing, routing,...



Cloud Delivery Models



The Cloud Foundry Stack

garden-linux or garden-runc containers

routing, authentication, app storage & exec., app lifecycle, services, messaging, metrics & logging

CF (&Diego)

Windows Server 2012

ubuntu 14.04 LTS

CentOS 7

vmware

amazon web services

Google Cloud Platform

Microsoft Azure

openstack

Platform-as-a-Service

- Applications
- Data
- Runtime
- Middleware
- OS
- Virtualization
- Servers
- Storage
- Networking

Dev vs Ops

`cf push APP`

`cf bind-service APP MYSQL`

Buildpack takes care of runtime

`bosh target ENDPOINT`

`bosh deployment cf.yml`

`bosh deploy`

bosh is a tool for release, deployment and lifecycle management.

How to deploy an app/microservice

- Developer pushes the application / microservice
- CF selects a proper buildpack (if not specified for the app)
- Buildpack creates the eventual deployment artifact
  - Droplet that contains everything needed to run the app
  - Binaries, 3rd party res, libraries, environment variables, …
  - Binds services to app and expose via environment variables
    - MySql, MongoDB, other application services
- Built droplet will then being launched on platform
  - Droplet runs isolated in garden containers
  - Garden containers run on dedicated VMs only (Diego-cells)

Enable Dynatrace via CF service

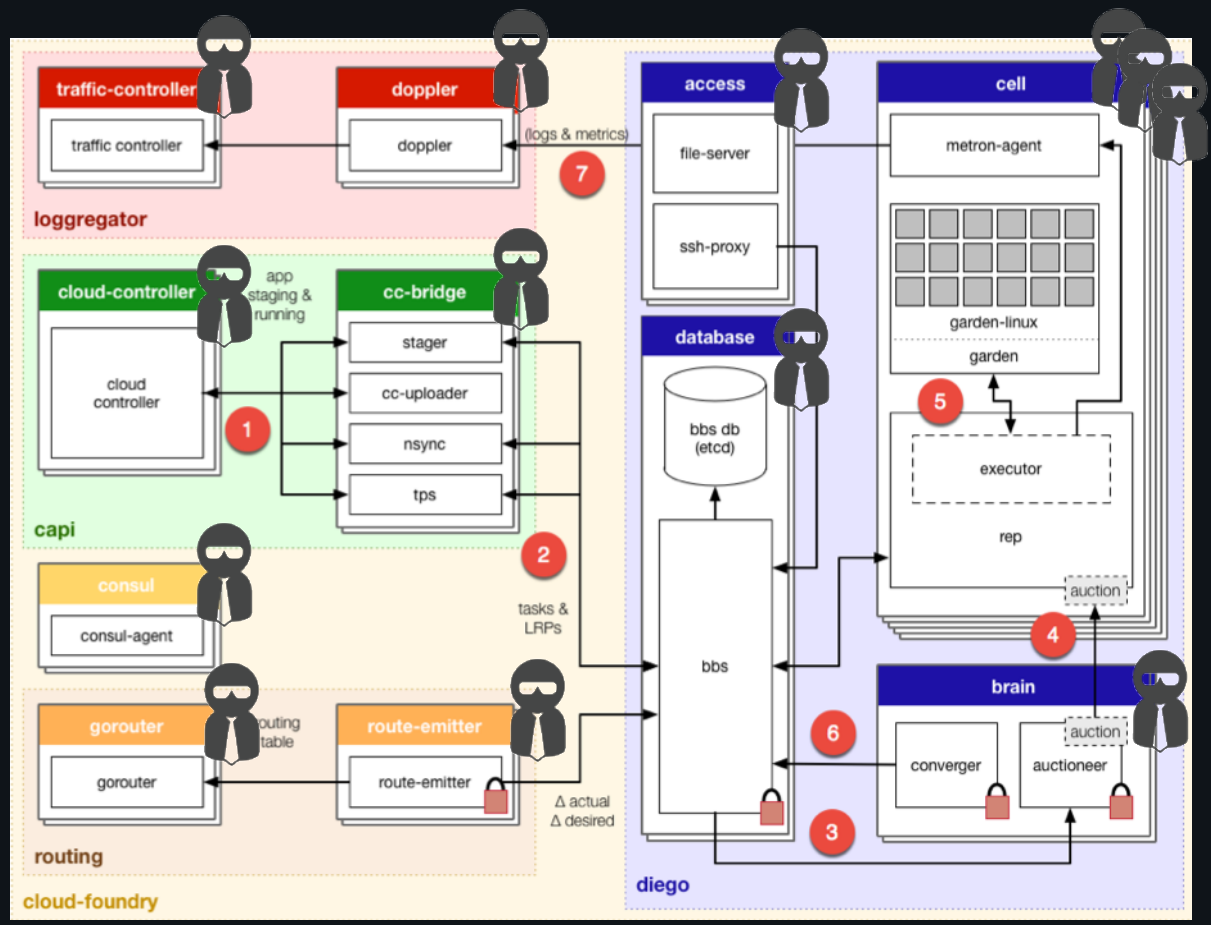
How to monitor CF apps with Dynatrace (PaaS approach)

- Injection of OneAgent by means of tech-specific buildpacks
  - Java buildpack — integrated with default buildpack
  - IBM Websphere Liberty Buildpack — integrated with default buildpack
  - Nodejs — via npm module
  - .Net (for Windows only) — co-deploy with app
  - PHP buildpack — prototype

Full-Stack Cloud Foundry Monitoring

- Deploying full OneAgent to each CF related VM
  - Via bosh-addon
- Auto-injection into garden containers (similar to Dynatrace Docker support)
- No buildpack integrations or additional services needed
- Complete CF cluster in one Dynatrace environment

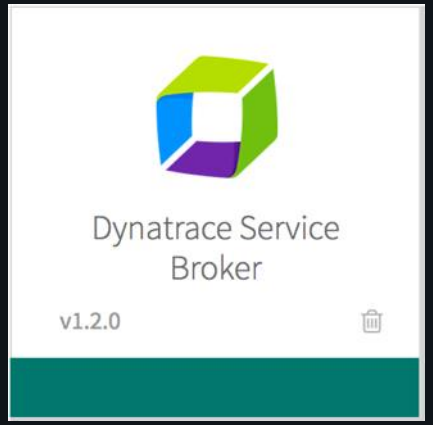
- Requirements:
  - CF architecture already based on garden-runc containers (present in CF releases running on stemcells based on 4.4 kernel)
  - Full access to host



Flavors of Cloud Foundry

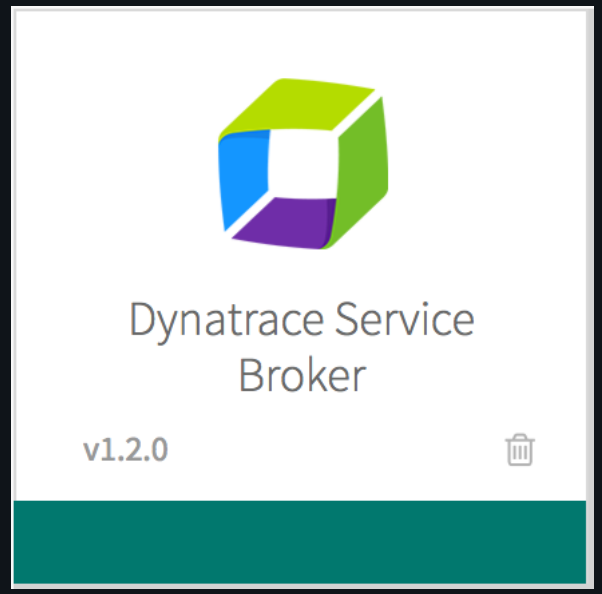
Pivotal Cloud Foundry

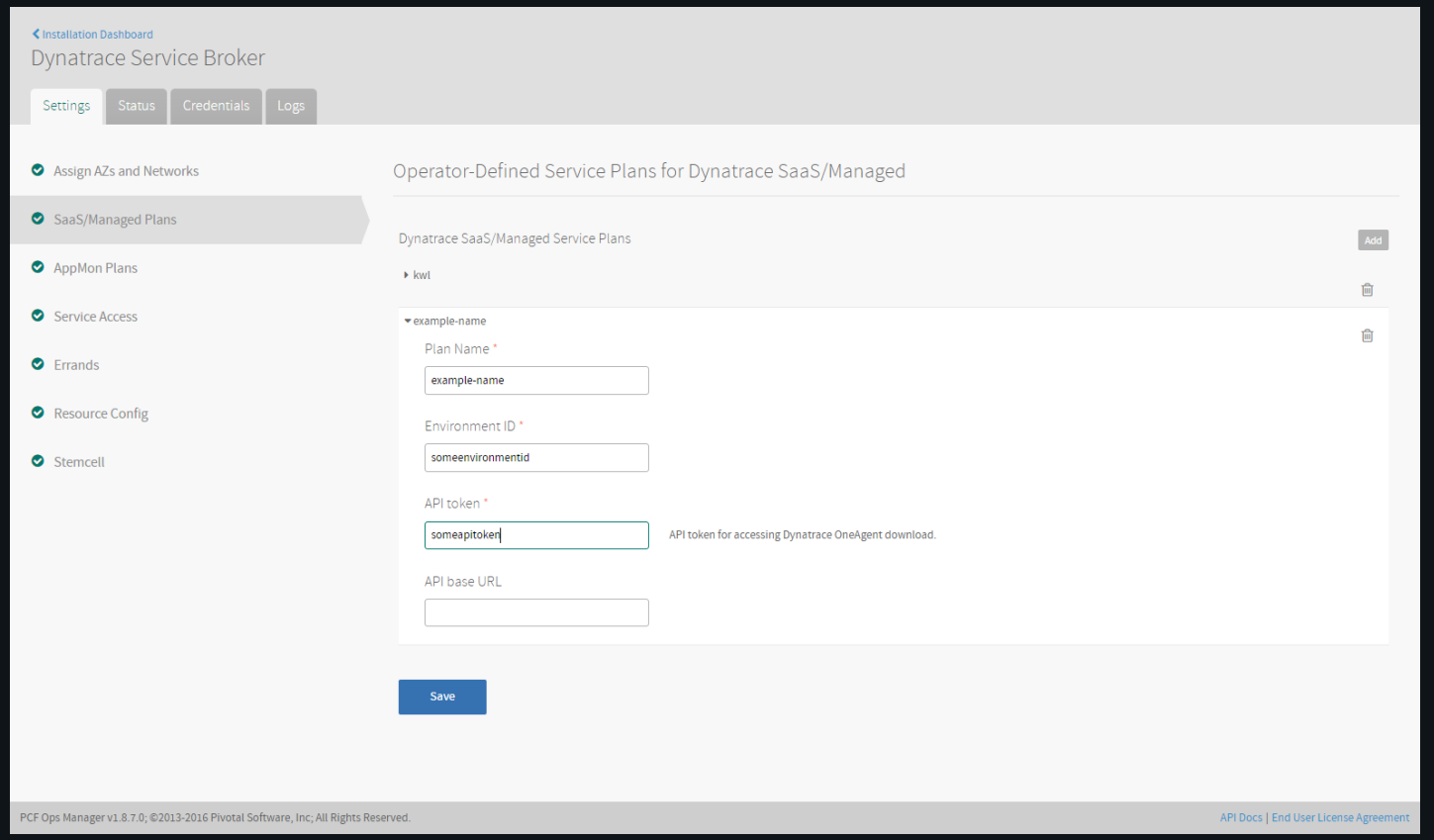
- Cloud Foundry is open source and maintained on GitHub
- Pivotal is a core contributor to Cloud Foundry projects
- Pivotal provides additional service & support packages to Cloud Foundry
  - Pivotal OpsManager (manage/deploy the environment)
  - Pivotal AppsManager (manage/deploy apps)
  - Pivotal Network (Marketplace to install "tiles")
- Deployment options
  - On-premise
  - SaaS (→ PWS, https://run.pivotal.io/)



Pivotal Cloud Foundry Tile

- Tile uses service-broker to manage various Dynatrace environments
- Central point of configuration – use it everywhere



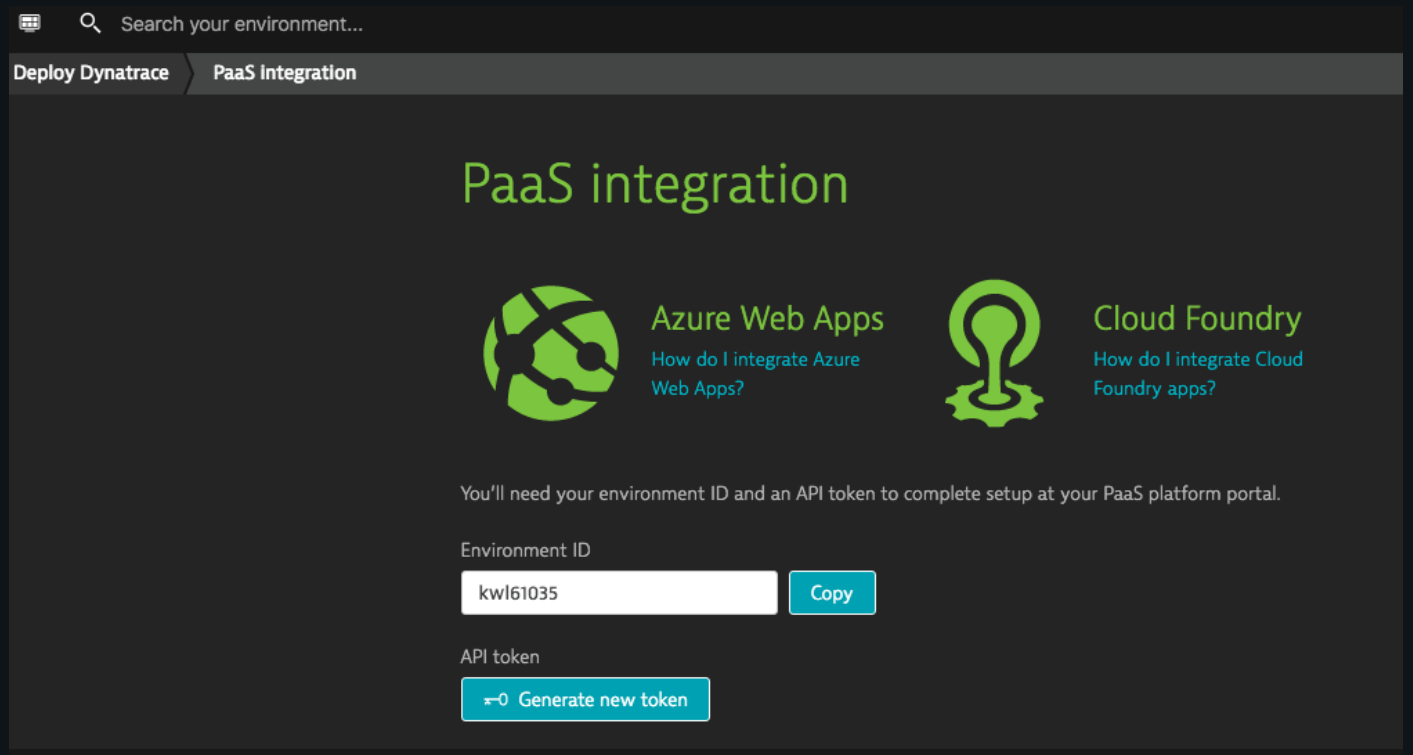


Other Distributions…

Deploying Applications with Dynatrace

Installing in Cloud Foundry

- Step 1
  - Generate PaaS Token



Installing in Cloud Foundry

- Step 2
  - Get the Cloud Foundry CLI

  - The code examples below use the ‘cf’ CLI for interfacing with Cloud Foundry clusters. You can find the appropriate CLI for your operating system on GitHub.

Installing in Cloud Foundry

- Step 3
  - Create a Dynatrace service in your Cloud Foundry environment

  - Option 1: Create a user-provided service
  - Option 2: Create a service instance via service broker
    - If you want to maintain Dynatrace credentials in a central location, use a service broker. For complete details, visit GitHub. You must configure and run the broker as an application, add the service broker to Cloud Foundry, enable service access to users, and finally, create the service instance.
  - Option 3: Create a service instance via the service broker on the Pivotal Network
    - If you run a Pivotal Cloud Foundry environment you can make use of the Dynatrace Service Broker tile on the Pivotal Network. For full details, see the instructions for the Dynatrace Service Broker tile.

Installing in Cloud Foundry

- Step 4
  - Bind Dynatrace service to your application

```
---
applications:
- name: spring-music
  memory: 768M
  instances: 1
  host: spring-music-${random-word}
  path: spring-music.war
  buildpack: https://github.com/cloudfoundry/java-buildpack.git
services:
  - dynatrace-service
```

  - You can either bind the created Dynatrace service to your application in your manifest.yml file prior to starting your app or you can bind the service to your app and restage the app afterwards.

  - If you're pushing a Node.js app to Cloud Foundry you need to first install the NPM module and load the module in your application.

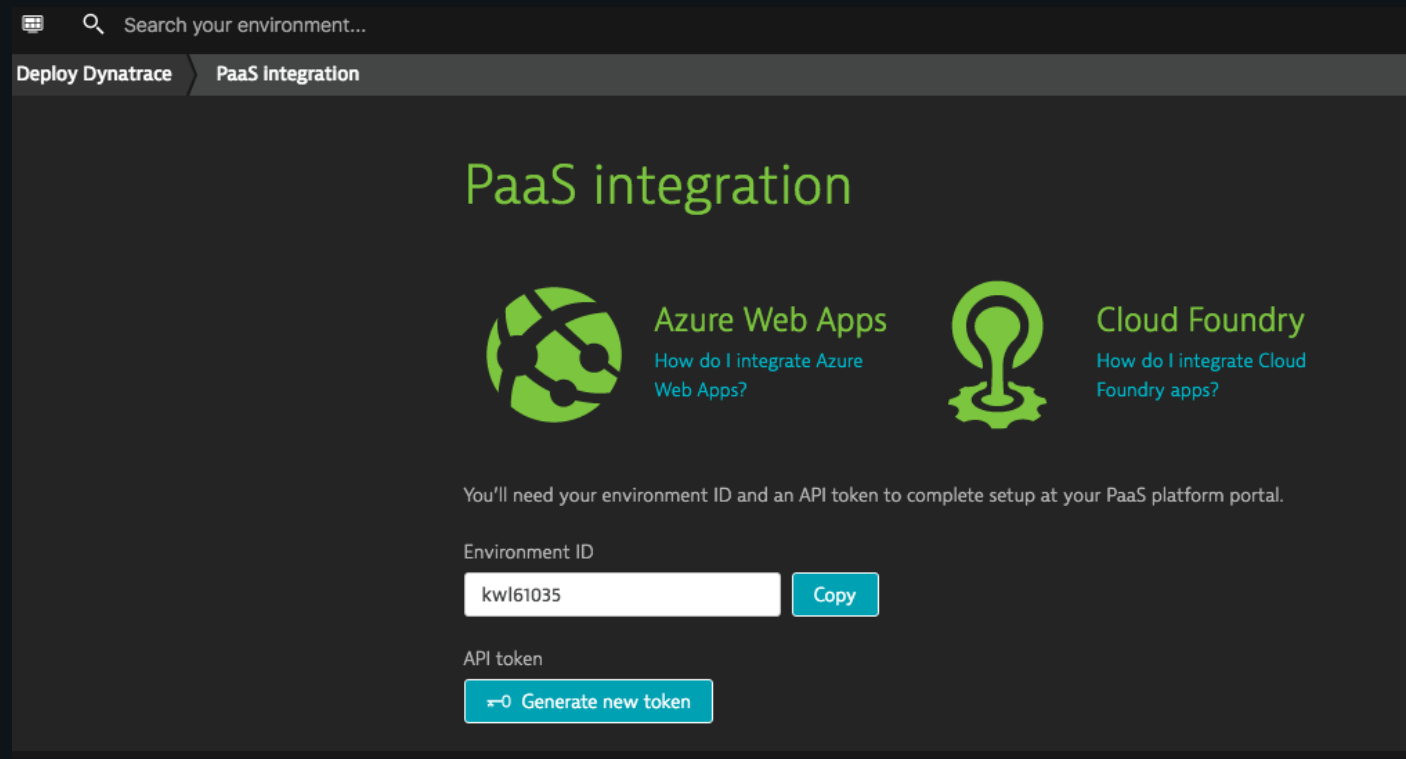
Deploying Full-Stack Monitoring for CloudFoundry

Installing the Full-Stack agent

- Dynatrace Full-Stack agent for CloudFoundy is deployed as a BOSH add-on
- To get started you need:
  - Dynatrace environment access
  - BOSH CLI access
  - CloudFoundry with Garden-runC v1.0.0 or higher
  - Enable Garden container monitoring in Monitored Technologies

Installing Full-Stack OneAgent in Cloud Foundry

- Step 1
  - Generate PaaS Token



Installing Full-Stack OneAgent in Cloud Foundry

- Upload and deploy the BOSH add-on
- Get the link to the latest version:
  - RELEASE=$(curl -s https://api.github.com/repos/dynatrace/bosh-oneagent-release/releases/latest | jq -r ".assets[] | select(.name) | .browser_download_url")
- Upload the addon to BOSH Director
  - For Ops Manager v1.10 or earlier:
    - bosh upload release PATH-TO-BINARY/dynatrace-release.tar.gz
  - For Ops Manager v1.11 or later:
    - bosh2 -e my-env upload-release PATH-TO-BINARY/dynatrace-release.tar.gz

Installing Full-Stack OneAgent in Cloud Foundry

- Update your runtime configuration to include the OneAgent
- Add a Dynatrace OneAgent Manifest – Example
- Note – you will need two job sections, one for Windows, one for Linux

```
releases:
- name: dynatrace-oneagent
  version: 1.0.3
  #specify version of latest release

addons:
- name: dynatrace-oneagent-addon
  jobs:
  - name: dynatrace-oneagent
    release: dynatrace-oneagent
  include:
    stemcell:
    - os: ubuntu-trusty
  exclude:
    jobs:
    - {name: smoke-tests, release: cf}
    - {name: push-apps-manager, release: push-apps-manager-release}
    - {name: deploy-notifications, release: notifications}
    - {name: deploy-notifications-ui, release: notifications-ui}
    - {name: push-pivotal-account, release: pivotal-account}
```

Installing Full-Stack OneAgent in Cloud Foundry

- Update BOSH Director runtime config and deploy changes
  - For Ops Manager v1.10 or earlier:
    - bosh update runtime-config PATH/runtime-config-dynatrace.yml
  - For Ops Manager v1.11 or later:
    - bosh2 -e my-env update-runtime-config PATH/runtime-config-dynatrace.yml
- This will apply to all new deployments going forward
- Deploy changes to apply the change across CloudFoundry
  - For Ops Manager v1.10 or earlier:
    - bosh deploy
  - For Ops Manager v1.11 or later:
    - bosh2 -e my-env -d deployment deploy

Useful Links

- https://run.pivotal.io/ Pivotal Web Services (SaaS)
- https://github.com/cloudfoundry/bosh-lite Run a local CF in a single VM (very unstable)
- https://github.com/cloudfoundry-samples/spring-music Spring-music (standard testing app)
- https://github.com/dynatrace-innovationlab/dynatrace-service-broker Simple service broker
- https://github.com/cloudfoundry/java-buildpack Java Buildpack with Dynatrace included
- https://github.com/cloudfoundry/ibm-websphere-liberty-buildpack Liberty BP w/ Dynatrace
- https://help.dynatrace.com/monitor-paas-environments/cloudfoundry/how-do-i-monitor-cloudfoundry/
- https://help.dynatrace.com/monitor-paas-environments/cloudfoundry/how-do-i-monitor-bluemix-cloudfoundry/

Demo

Kubernetes

Monitoring Kubernetes with Dynatrace

- Kubernetes is a next generation platform for deploying and running containerized applications at scale
- Dynatrace OneAgent is container-aware and comes with built-in support for out-of-the-box whitebox monitoring of Kubernetes
- Dynatrace supports full-stack monitoring for Kubernetes
- However, if you don't have access to the infrastructure layer, Dynatrace provides also the option of application-only monitoring

Kubernetes Full-Stack Monitoring

- From the application down to the infrastructure layer
- Preferred method for K8s that allow running privileged containers on nodes
- Out of the box, automated cluster and workload monitoring
- Auto-injection of code modules into pods for full-stack monitoring
- Rolled out via OneAgent Operator or DaemonSet
- Preferred for:
  - Google GKE
  - Azure AKS
  - Amazon EKS
  - Pivotal PKS
  - Other 'vanilla' K8s Clusters

Deploying with a DaemonSet (Full-Stack)

- Required for K8s < 1.8
- Steps:
    1. Download or copy the dynatrace-oneagent.yml Kubernetes template
    2. Deploy Dynatrace OneAgent using the created file dynatrace-oneagent.yml
    3. Verify that the dynatrace-oneagent DaemonSet has deployed pods to the cluster nodes successfully
- Installation is easy and straightforward but OneAgent updates can be cumbersome (day 2 operations)
- If you have K8s > 1.8 you should use the OneAgent Operator

Deploying with OneAgent Operator (Full-Stack)

- Recommended approach for K8s > 1.8
- Operators introduced in K8s v1.7 to allow community to implement domain-specific applications as 1st class K8s objects in a cloud-native style
- Dynatrace makes use of the Operator concept to ease 'Day 2' operations: management, updates, and roll-outs of new OneAgent versions
- Value added:
  - Fine-grained control of roll-outs
  - Updates can be performed automatically
  - Pods can be recycled to apply latest updates automatically

Kubernetes Application-Only Monitoring

- Designed for locked-down K8s based environments with no infra access
- Provides visibility on a container image basis (no cluster node visibility)
- OneAgent code modules are integrated with container images
- Universal injection of code modules
- Rolled out as part of normal K8s workloads
- Prefered method for:
  - Google App Engine
  - Amazon ECS Fargate
  - Other locked-down K8s based platforms
- Several options available...

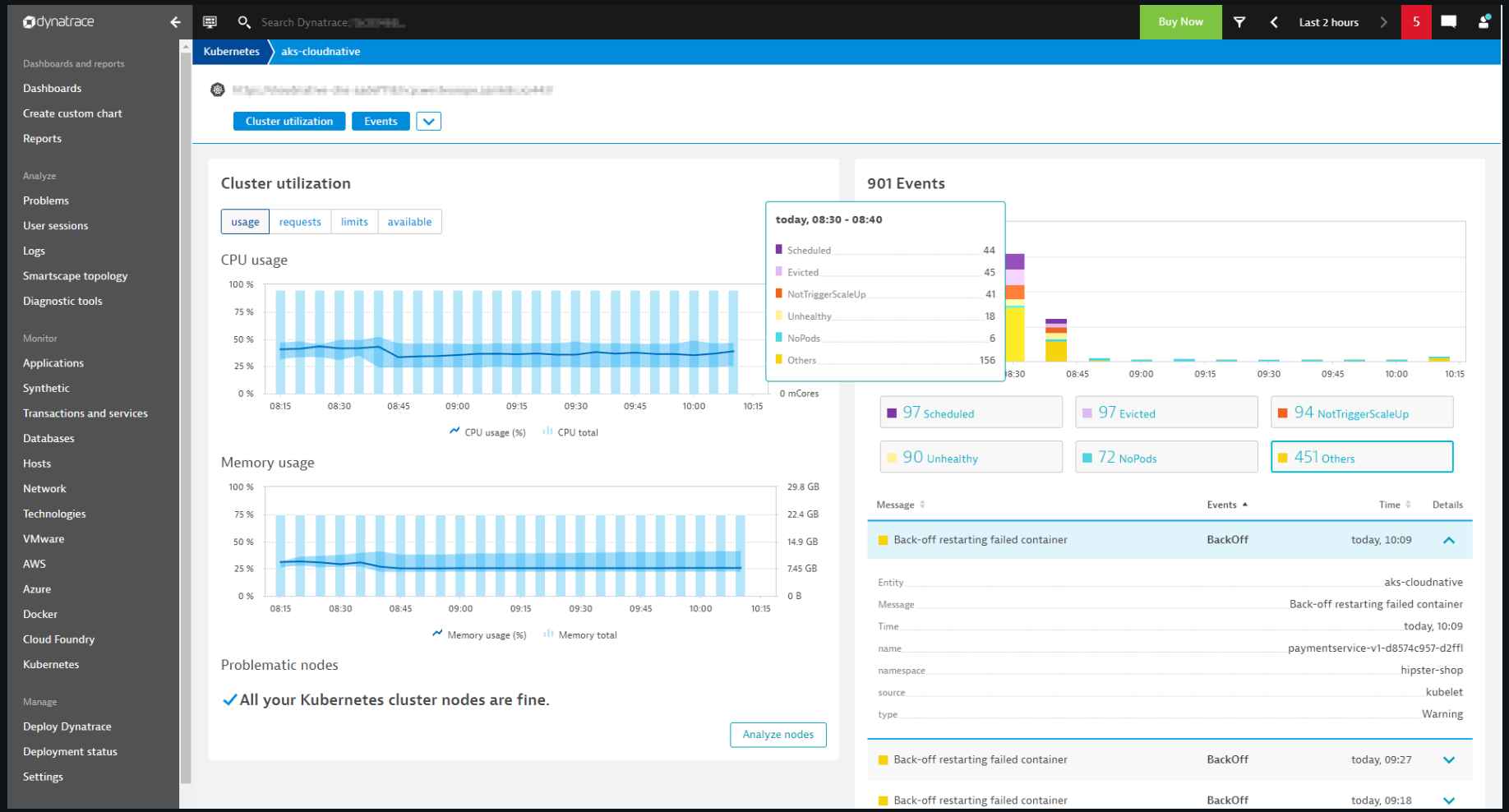
Kubernetes Application-Only Monitoring Strategies

Application-only strategy	Overview	Advantages
Automated injection	OneAgent Operator injects into pods as they start.	Injection is centrally managed. Pods can be selected by using namespaces or pod-level annotations.
Pod runtime injection	Pod uses an init container to download OneAgent.	Can mount a volume to speed up injection for subsequent pods.
Container build-time injection	Copies image layer into Docker image during build process.	No dependency on specific k8s features like init containers or admission controllers.

Further Kubernetes Integrations

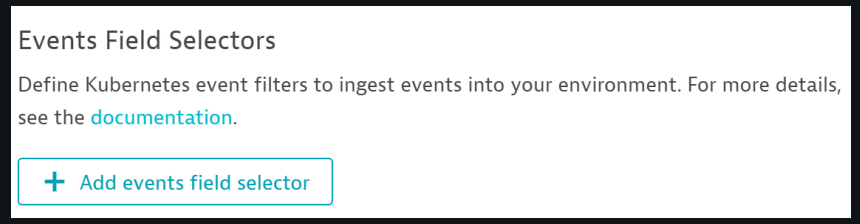
- Configure Istio for OneAgent traffic in K8s
  - Istio is a service mesh that helps running distributed microservice architectures
  - Uses the sidecar pattern to deploy a proxy to pods which then intercepts network traffic between your microservices
  - Must configure Istio to enable egress traffic to your Dynatrace environment
- Leverage tags defined in Kubernetes deployments
  - Dynatrace automatically derives tags from your Kubernetes/OpenShift labels
  - It's recommended that you define additional metadata at the deployed system
    - You can use K8s or OpenShift annotations
  - Can be used for automated tagging rules
- Monitor your Kubernetes Cluster Health
  - Dedicated built-in Kubernetes/OpenShift cluster overview pages that provides you with extended visibility into Kubernetes cluster performance and health.

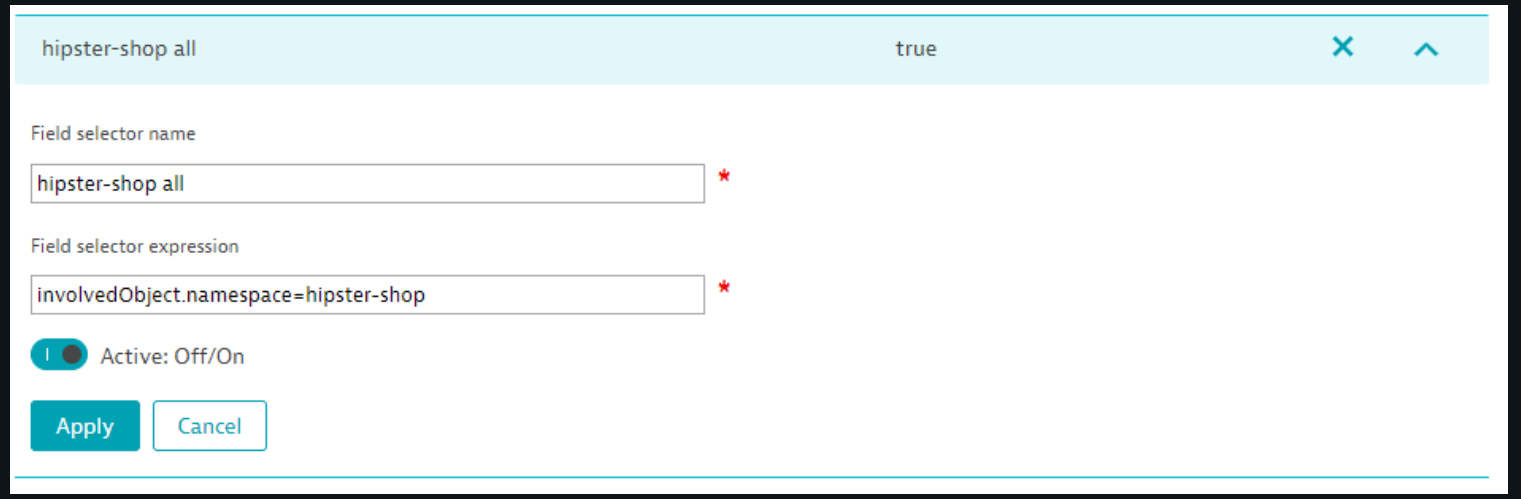
Kubernetes Cluster Monitoring + Native Events



Kubernetes Cluster Monitoring + Native Events

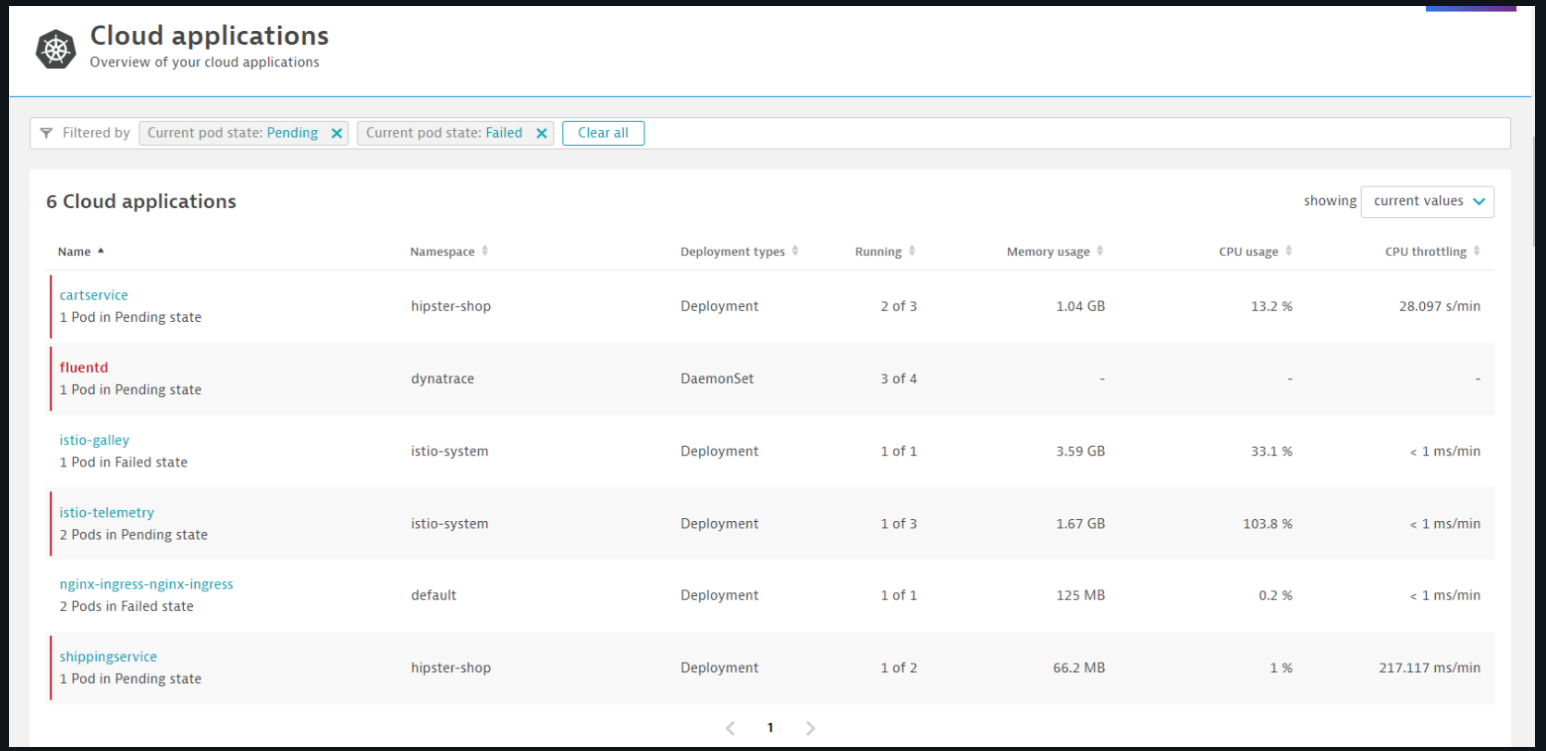
- Settings -> Cloud and Virtualization -> Kubernetes
- You can enable event integration on the Kubernetes settings page where you set up Kubernetes API monitoring for your clusters.
- Event ingestion follows the Kubernetes established format of field selectors. With this approach, we achieve our goal of flexibility by choosing events based on event resource fields such as `source.component`, `type`, or `involvedObject`.





Pod + Namespace Oriented Monitoring

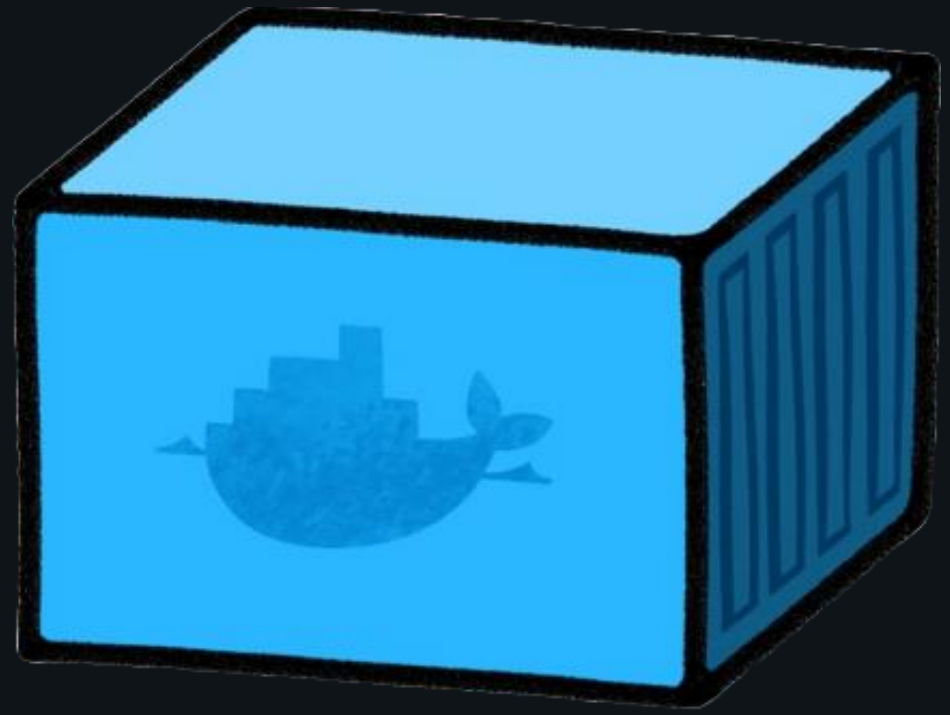
- Identify failed and pending pods
- Find resource intensive pods
- Many pod and namespace oriented metrics for dashboards + alerting



OpenShift

OpenShift v2 vs. v3

Cartridges (v2)



Docker Containers (v3)

OpenShift Architecture

You provide

Applications and Data

DevOps Tools & User Experience

Language Runtimes, Middleware, Databases & Other Services

Container Orchestration & Management

Container API

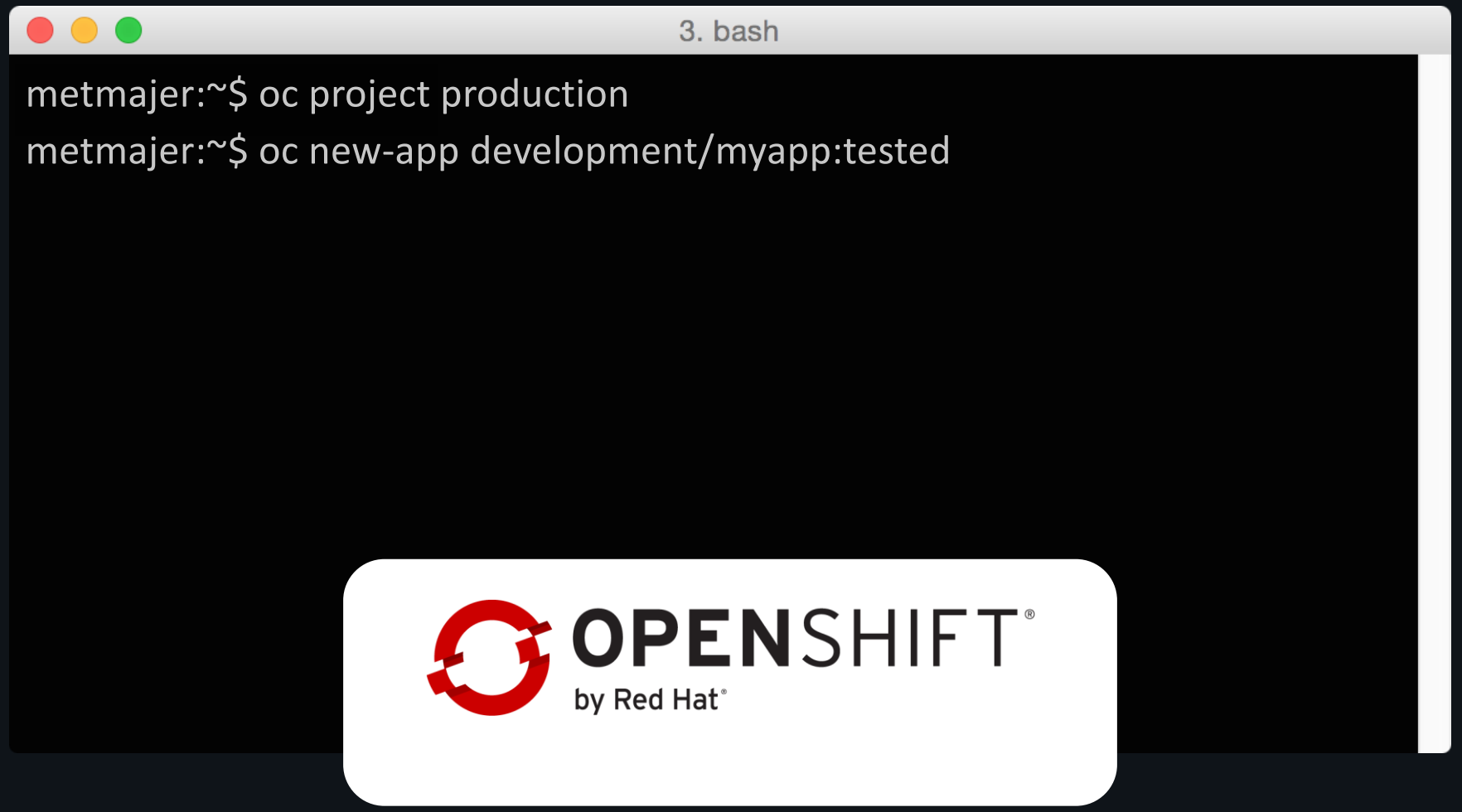
Container Host

You provide

Infrastructure

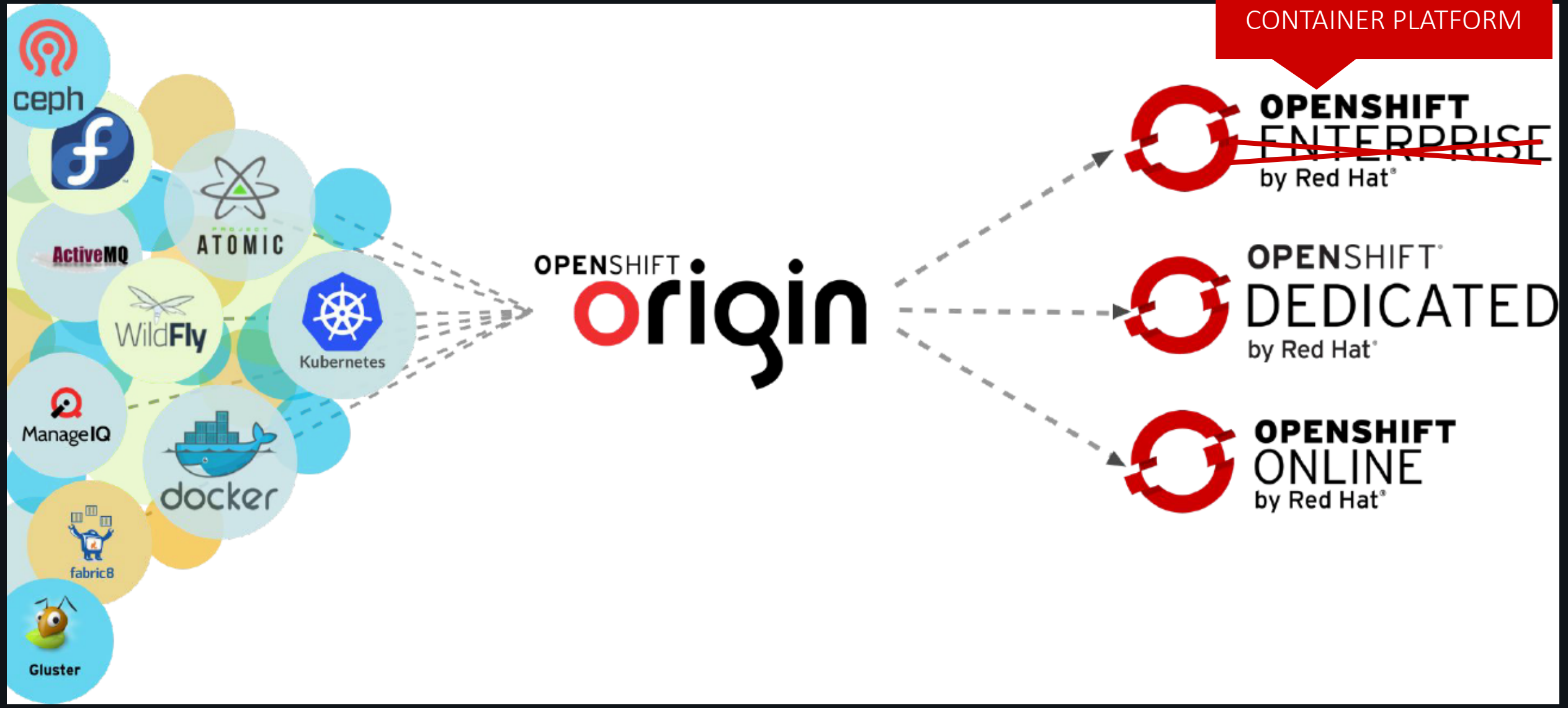
On-premise or cloud-based (Azure, EC2, GCE, etc.)

How to promote an app into production today…



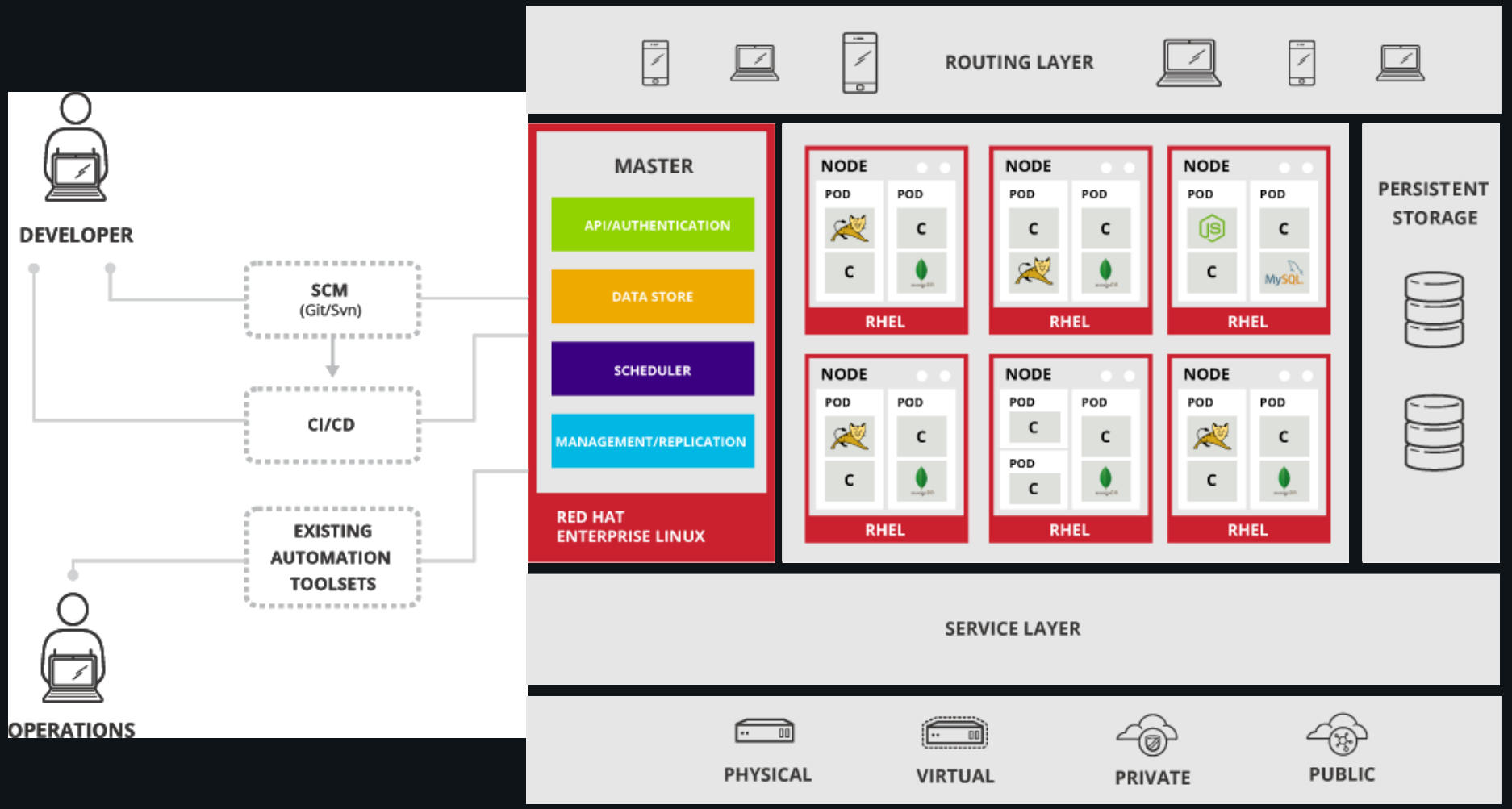
OpenShift = Docker + Kubernetes + UX

Community Powered Innovation



OpenShift Architecture

OpenShift Architecture



How do I...

…deploy OneAgent?

Option A: Full Stack via OneAgent Operator

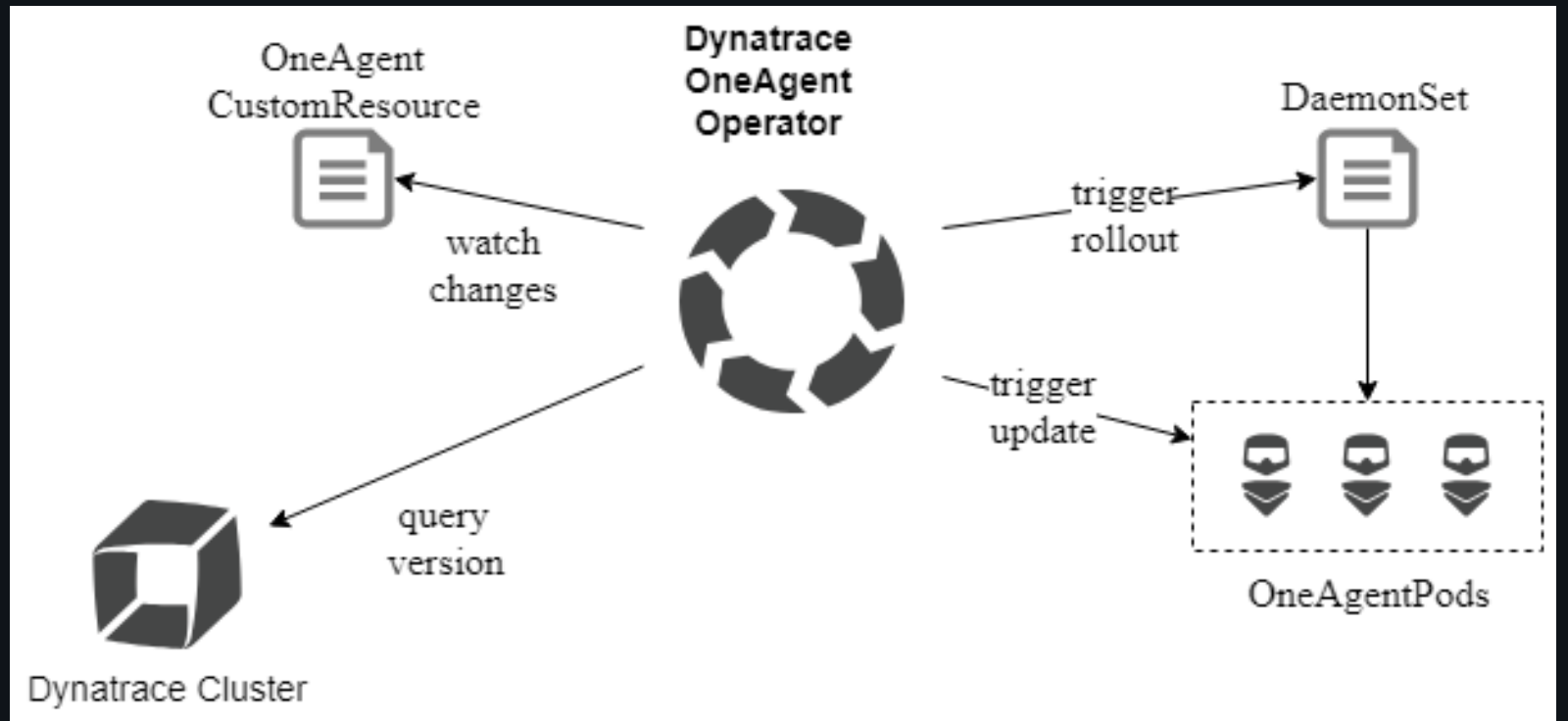
- What is an operator in Kubernetes?
  - Kubernetes version 1.7 introduced the concept of custom resources and controllers, which allow for extending the Kubernetes API. These extension capabilities enable the Kubernetes community to implement domain-specific applications as first-class Kubernetes objects in a cloud-native style. This means you can define the desired state of workloads in a declarative manner and create custom controller logic that takes continuous action to achieve and maintain the desired state.
- Dynatrace OneAgent Operator for 'Day 2' operations
  - Dynatrace makes use of the Operator concept by putting operational knowledge into software and automating the management, updates, and roll-outs of new Dynatrace OneAgent versions.

Value-add of Dynatrace OneAgent operator

- OneAgent Operator offers fine-grained control of OneAgent roll-outs. You can thereby select nodes based on node labels. This enables you to monitor selected nodes using different Dynatrace environments. OneAgent Operator also supports tolerations so you can deploy OneAgent on tainted nodes.
- Dynatrace OneAgent updates are performed automatically, as soon as they’re available. When pending updates are available, OneAgent Operator takes care of recycling all pods that haven't yet picked up the latest version.
- OneAgent Operator ensures that you always monitor your OpenShift cluster with the latest OneAgent version.

How does OneAgent Operator Work?

- Dynatrace OneAgent Operator registers itself as a controller that watches for resources of type **`OneAgent`** as defined by a CustomResourceDefinition. This allows you to define a configuration that describes your OneAgent deployment. By loading the configuration into Kubernetes or OpenShift, the configuration is automatically passed to the custom controller which ensures the rollout of OneAgent based on your specification.



Depoly OneAgent via Operator (Openshift 3.9 or higher)

- Prerequisites
  - Get an API token for the Dynatrace API. This token is later referenced as `API_TOKEN`.
  - Get a Platform-as-a-Service token. This token is later referenced as `PAAS_TOKEN`.
- Install OneAgent operator
  - Create the necessary objects for OneAgent Operator. OneAgent Operator acts on its separate project `dynatrace`. It holds the operator deployment and all dependent objects like permissions, custom resources and the corresponding DaemonSet.
    - $ LATEST_RELEASE=$(curl -s https://api.github.com/repos/dynatrace/dynatrace-oneagent-operator/releases/latest | grep tag_name | cut -d '"' -f 4)
    - $ oc create -f https://raw.githubusercontent.com/Dynatrace/dynatrace-oneagent-operator/$LATEST_RELEASE/deploy/openshift.yaml
    - $ oc -n dynatrace logs -f deployment/dynatrace-oneagent-operator

OneAgent Operator install continued

- Create the secret holding API and PaaS tokens for authenticating to the Dynatrace cluster. The name of the secret is important in a later step when you configure the custom resource (`.spec.tokens`). In the following code-snippet the name is **`oneagent`**. Be sure to replace **`API_TOKEN`** and **`PAAS_TOKEN`** with the values explailned in the previous slide.
  - $ oc -n dynatrace create secret generic oneagent --from-literal="apiToken=API_TOKEN" --from-literal="paasToken=PAAS_TOKEN"
- Save the following custom resource snippet to a file **`cr.yaml`**. The rollout of Dynatrace OneAgent is governed by a custom resource of type **`OneAgent`**.
  - Find snippet in Dynatrace Doc Here
  - Alternatively, you can use the snippet from the GitHub repository.
    - `$ curl -o cr.yaml https://raw.githubusercontent.com/Dynatrace/dynatrace-oneagent-operator/$LATEST_RELEASE/deploy/cr.yaml`
    -
    -

OneAgent Operator install continued

- Adapt the values of the custom resource as indicated in the following table.

Parameter	Description	Default value
`apiUrl`	Dynatrace SaaS: Replace `ENVIRONMENTID` with your Dynatrace environment ID in `https://ENVIRONMENTID.live.dynatrace.com/api`. Dynatrace Managed: Provide your Dynatrace Server URL (`https://<YourDynatraceServerURL>/e/<ENVIRONMENTID>/api`)	
`tokens`	Name of the secret that holds the API and PaaS tokens from above.	Name of custom resource (`.metadata.name`) if unset
`image`	Define the OneAgent image to be taken. Defaults to the publicly available OneAgent image on Docker Hub. In order to use the certified OneAgent image from Red Hat Container Catalog you need to set `.spec.image` to `registry.connect.redhat.com/dynatrace/oneagent` in the custom resource and provide image pull secrets as shown in the next step.	`docker.io/dynatrace/oneagent:latest` if unset
`args`	Parameters to be passed to the OneAgent installer. All the command line parameters of the installer are supported, with the exception of `INSTALL_PATH`. We recommend to set `APP_LOG_CONTENT_ACCESS=1`	[]

OneAgent Operator Install Continued

- Provide the image pull secret in case you set `.spec.image` to `registry.connect.redhat.com/dynatrace/oneagent`.
  - $ oc -n dynatrace create secret docker-registry redhat-connect --docker-server=registry.connect.redhat.com --docker-username=REDHAT_CONNECT_USERNAME --docker-password=REDHAT_CONNECT_PASSWORD --docker-email=unused
  - $ oc -n dynatrace create secret docker-registry redhat-connect-sso --docker-server=sso.redhat.com --docker-username=REDHAT_CONNECT_USERNAME --docker-password=REDHAT_CONNECT_PASSWORD --docker-email=unused
  - $ oc -n dynatrace secrets link dynatrace-oneagent redhat-connect --for=pull
  - $ oc -n dynatrace secrets link dynatrace-oneagent redhat-connect-sso --for=pull
- Create the custom resource
  - $ oc create -f cr.yaml

Updates

- OneAgent Operator automatically takes care of the lifecycle of the deployed OneAgents, so you don't need to update OneAgent pods yourself.
- Update OneAgent Operator
  - Please review the release notes of the Operator for any breaking changes of the custom resource. If the custom resource of the new version is compatible with the already deployed one, you can simply set the OneAgent Operator image to the new tagged version. Be sure to replace **`vX.Y.Z`** with the new version in the following command.
  - $ oc -n dynatrace set image deployment dynatrace-oneagent-operator *=quay.io/dynatrace/dynatrace-oneagent-operator:vX.Y.Z

Depoly OneAgent via Operator (Kubernetes 1.9 or higher)

- Prerequisites
  - Get an API token for the Dynatrace API. This token is later referenced as **API_TOKEN**.
  - Get a Platform-as-a-Service token. This token is later referenced as **PAAS_TOKEN**.
- Install OneAgent operator
  - Create the necessary objects for OneAgent Operator. OneAgent Operator acts on its separate namespace **dynatrace**. It holds the operator deployment and all dependent objects like permissions, custom resources and the corresponding DaemonSet.
    - $ LATEST_RELEASE=$(curl -s https://api.github.com/repos/dynatrace/dynatrace-oneagent-operator/releases/latest | grep tag_name | cut -d '"' -f 4)
    - $ kubectl create -f https://raw.githubusercontent.com/Dynatrace/dynatrace-oneagent-operator/$LATEST_RELEASE/deploy/kubernetes.yaml
    - $ kubectl -n dynatrace logs -f deployment/dynatrace-oneagent-operator

OneAgent Operator install continued (Kubernetes 1.9 or higher)

- Create the secret holding API and PaaS tokens for authenticating to the Dynatrace cluster. The name of the secret is important in a later step when you configure the custom resource (`.spec.tokens`). In the following code-snippet the name is **oneagent**. Be sure to replace **API_TOKEN** and **PAAS_TOKEN** with the values explained in the previous slide.
  - $ kubectl -n dynatrace create secret generic oneagent --from-literal="apiToken=API_TOKEN" --from-literal="paasToken=PAAS_TOKEN"
- Save the following custom resource snippet to a file **cr.yaml**. The rollout of Dynatrace OneAgent is governed by a custom resource of type **OneAgent**.
  - Find snippet in Dynatrace Doc Here
  - Alternatively, you can use the snippet from the GitHub repository.
    - `$ curl -o cr.yaml https://raw.githubusercontent.com/Dynatrace/dynatrace-oneagent-operator/$LATEST_RELEASE/deploy/cr.yaml`

OneAgent Operator install continued (Kubernetes 1.9 or higher)

- Adapt the values of the custom resource as indicated in the following table.
- Create the custom resource
  - $ oc create -f cr.yaml

Parameter	Description	Default value
`apiUrl`	Dynatrace SaaS: Replace `ENVIRONMENTID` with your Dynatrace environment ID in `https://ENVIRONMENTID.live.dynatrace.com/api`. Dynatrace Managed: Provide your Dynatrace Server URL (`https://<YourDynatraceServerURL>/e/<ENVIRONMENTID>/api`)	
`tokens`	Name of the secret that holds the API and PaaS tokens from above.	Name of custom resource (`.metadata.name`) if unset
`args`	Parameters to be passed to the OneAgent installer. All the command line parameters of the installer are supported, with the exception of `INSTALL_PATH`. We recommend to set `APP_LOG_CONTENT_ACCESS=1`	[]

Updates

- OneAgent Operator automatically takes care of the lifecycle of the deployed OneAgents, so you don't need to update OneAgent pods yourself.
- Update OneAgent Operator
  - Please review the release notes of the Operator for any breaking changes of the custom resource. If the custom resource of the new version is compatible with the already deployed one, you can simply set the OneAgent Operator image to the new tagged version. Be sure to replace **`vX.Y.Z`** with the new version in the following command.
  - $ kubectl -n dynatrace set image deployment dynatrace-oneagent-operator *=quay.io/dynatrace/dynatrace-oneagent-operator:vX.Y.Z

…deploy OneAgent?

Option B: Application-Only deployment

- You'll need the Dynatrace environment credentials and a generated PaaS Token.
- Integrate OneAgent into application images
  - Define variables with optional default values using **ARG** instructions, as shown in the code snippet below.
    - ARG DT_API_URL="https://<environmentID>.live.dynatrace.com/api"
    - ARG DT_API_TOKEN="<token>"
    - ARG DT_ONEAGENT_OPTIONS="flavor=default&include=<technology1>&include=<technology2>"
    - ENV DT_HOME="/opt/dynatrace/oneagent"

Option B Application-only deployment

- Add the following commands to your current Dockerfile to integrate OneAgent and activate instrumentation of your application.
  - ARG DT_API_URL="https://<environmentID>.live.dynatrace.com/api" ARG DT_API_TOKEN="<token>"
  - ARG DT_ONEAGENT_OPTIONS="flavor=default&include=<technology1>&include=<technology2>"
  - ENV DT_HOME="/opt/dynatrace/oneagent" RUN mkdir -p "$DT_HOME" && \ wget -O "$DT_HOME/oneagent.zip" "$DT_API_URL/v1/deployment/installer/agent/unix/paas/latest?Api-Token=$DT_API_TOKEN&$DT_ONEAGENT_OPTIONS" && \ unzip -d "$DT_HOME" "$DT_HOME/oneagent.zip" && \ rm "$DT_HOME/oneagent.zip"
  - ENTRYPOINT ["/opt/dynatrace/oneagent/dynatrace-agent64.sh"]
  - CMD ["executable", "param1", "param2"] # the command of your application, e.g. java

Option B Application-only deployment

- Build your application image
  - $ oc new-build --binary --strategy=docker --allow-missing-images yourapp
  - $ oc patch bc/yourapp --type=json --patch='[{"op":"remove","path":"/spec/strategy/dockerStrategy/from"}]'
  - $ oc start-build yourapp --from-dir=. --follow
- Updates:
  - Each time a new version of Dynatrace OneAgent becomes available, you must rebuild your applications' Docker image. Following restart, your OpenShift applications will be monitored with the latest version of OneAgent.
  - If you've specified a default OneAgent installation version for new hosts and applications using OneAgent updates settings, your OpenShift applications will be automatically monitored by the defined default version of Dynatrace OneAgent.

Cloud Application

Heroku

Public Beta – Heroku BuildPack

- The buildpack for Dynatrace OneAgent enables cloud-native monitoring of your Heroku application by integrating OneAgent into your application's slug and dyno – *without modification of your applications' source code*.
- Add the OneAgent buildpack as an additional buildpack.
  - heroku buildpacks:add https://github.com/Dynatrace/heroku-buildpack-dynatrace.git
- Value –add at a glance
  - Deep application monitoring and code-level details for Java, PHP, Node.js and more
  - Automatic root cause analysis
  - Insights into how your Heroku app use databases – including detailed metrics for each statement
  - Real user monitoring
  - Automated external and third-party service monitoring (REST APIs)
- Install Guide - https://github.com/Dynatrace/heroku-buildpack-dynatrace

Virtualization

VMware

VMware

- Once Dynatrace OneAgent is installed on a virtual host and process monitoring is activated, you'll be able to see what's happening in your operating system
  - Specifically, how your host-based processes behave and communicate
- If you depend on virtualization for your infrastructure, Dynatrace will give you insight into how your virtual machines affect the performance of your web applications

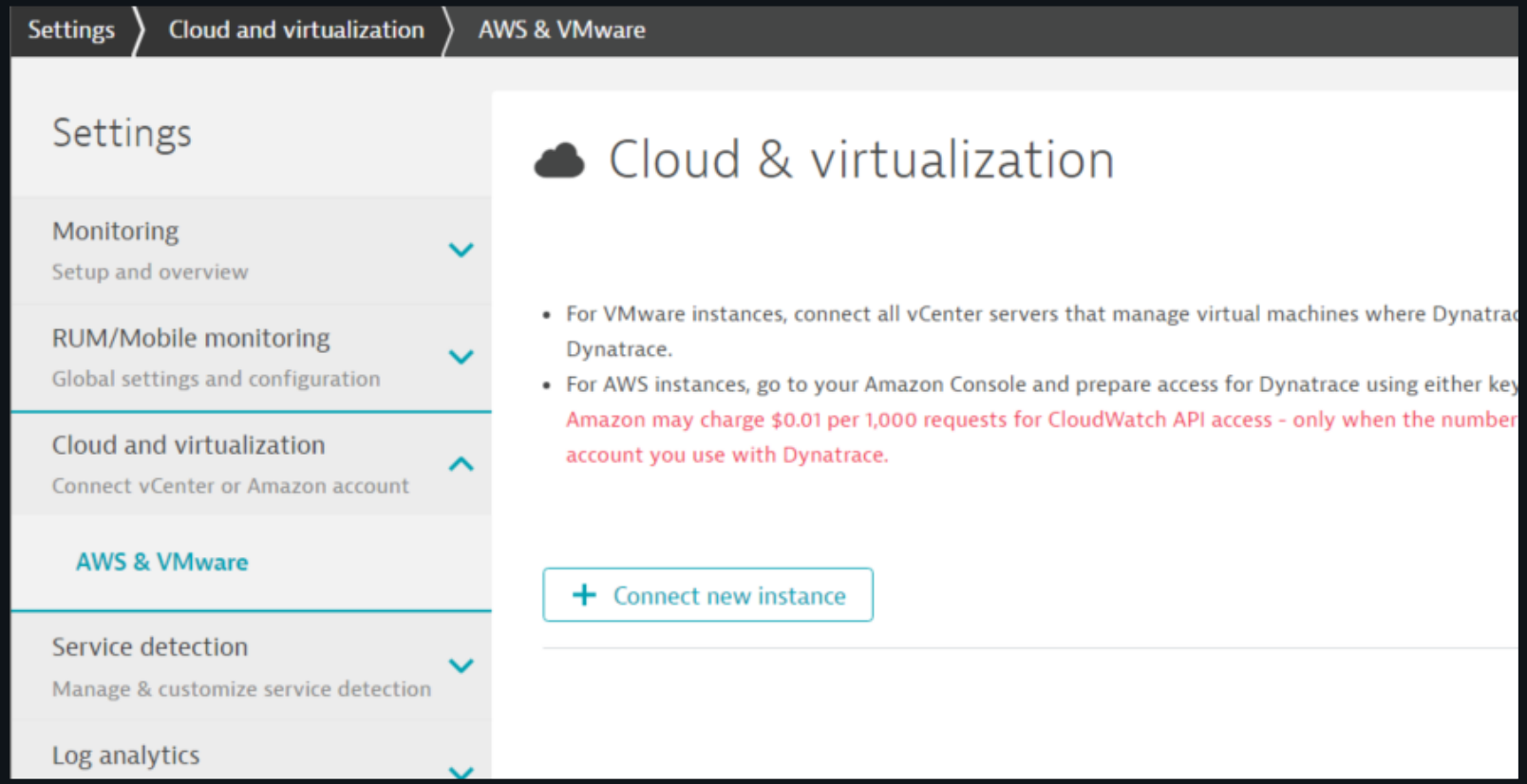
- If you want to take it one step further and monitor the virtualized infrastructure, follow the next slides to setup additional monitoring
  - Obtain crucial information related to any virtualized infrastructure you've deployed using VMware vCenter or standalone ESXi hosts

VMware Monitoring

- First you will need to setup an ActiveGate
- The AG will reach out remotely and collect metrics
- It relies on having read-only credentials and an open firewall

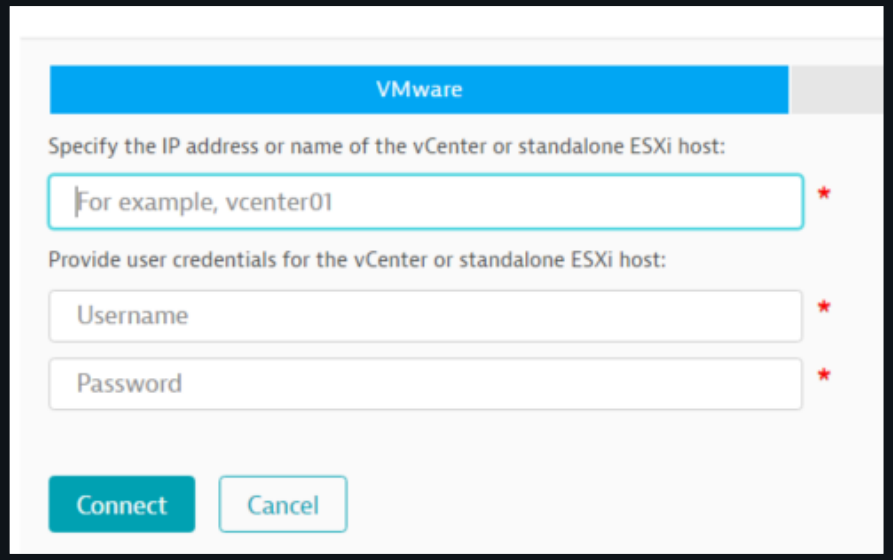
VMware Monitoring

- Second you will need to connect to your VMware platform
- This is done on the Settings → Virtualization page



VMware Monitoring

- Third you will decide which vCenter server or standalone ESXi host you want to monitor
- Enter corresponding user credentials so that the ActiveGate can log in and collect monitoring data
  - The required privileges for this user are **view and read-only access**
- Only add standalone ESXi hosts for monitoring
  - If your ESXi host is managed by a vCenter server, connect the vCenter server to Dynatrace



VMware Monitoring

- When the ActiveGate connects to your VMware platform for the first time, it collects information about all available virtual machines and matches it with the information it's received from all machines that have Dynatrace OneAgent installed on them

- Repeat this process for all other vCenter servers or standalone ESXi hosts in your environment, covering all virtualized systems that run Dynatrace OneAgent

Demo

Misc

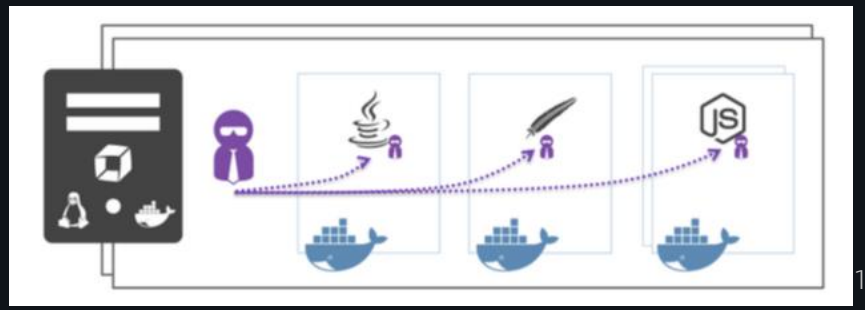
Docker

Can I monitor Docker?

- Dynatrace offers full-featured Docker monitoring, giving you all the same monitoring functionality for Dockerized applications that is available for non-containerized applications.

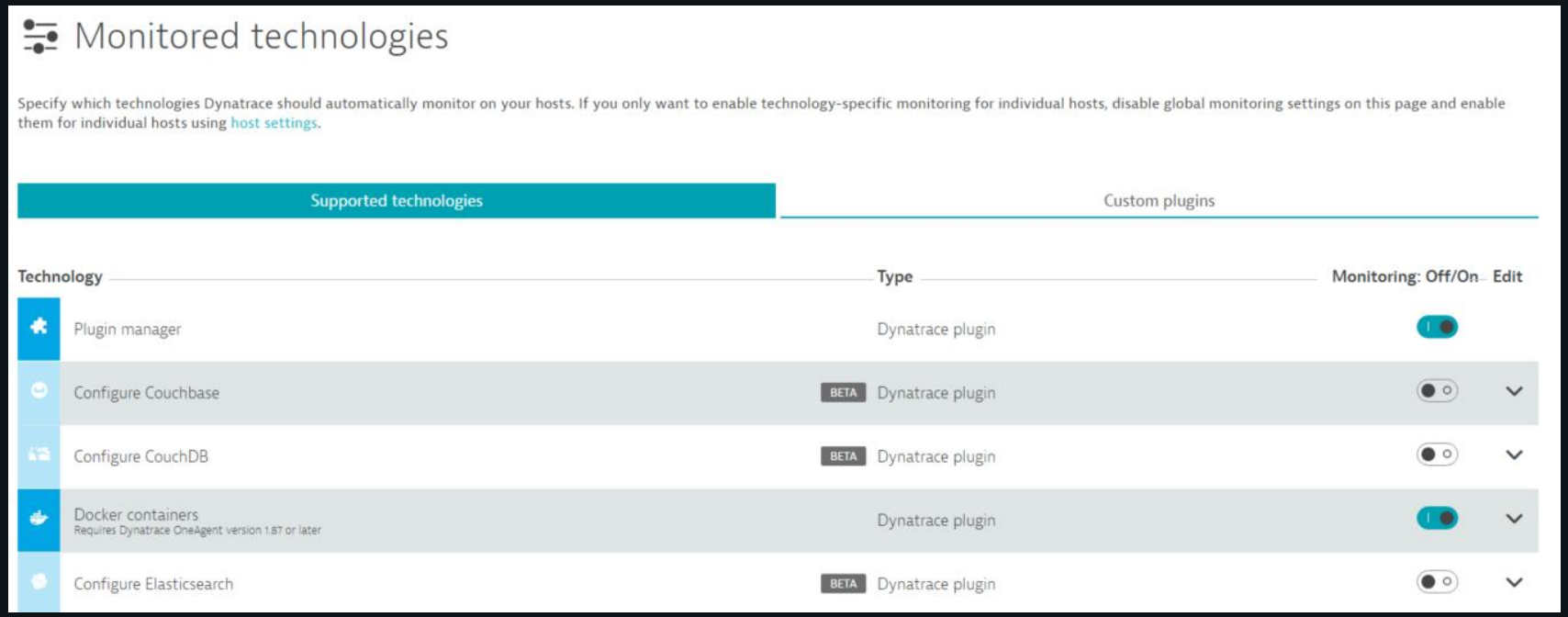
Get started with Docker

- Dynatrace seamlessly integrates with existing Docker environments and automatically monitors your containerized applications and services
- Dynatrace hooks into containers and provides code for injecting the agent into containerized processes
- There's no need to modify your Docker images, modify run commands, or create additional containers to enable Docker monitoring
- Simply install Dynatrace OneAgent on your hosts that serve containerized applications and services
- Dynatrace automatically detects the creation and termination of containers and monitors the applications and services contained within those containers



Enable Docker container monitoring

- Go to **Settings > Monitored Technologies**.
- Enable the **Docker containers** switch.
- If you haven't done so already, also enable the **Plugin manager** switch and the **Docker** switch further down the list.

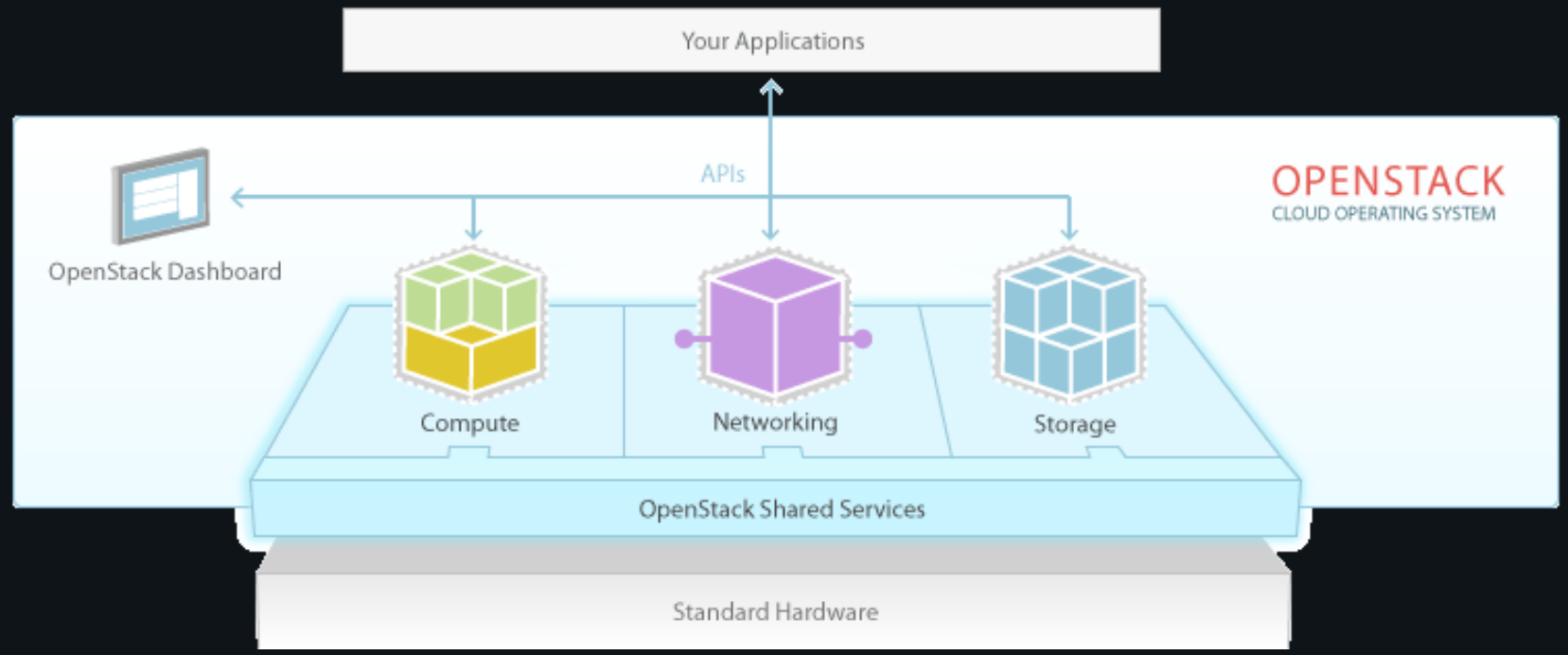


Demo

OpenStack

What is OpenStack

- Cloud operating system
- Controls large pools of resources
  - computing
  - networking
  - storage
- IaaS
- AWS at home

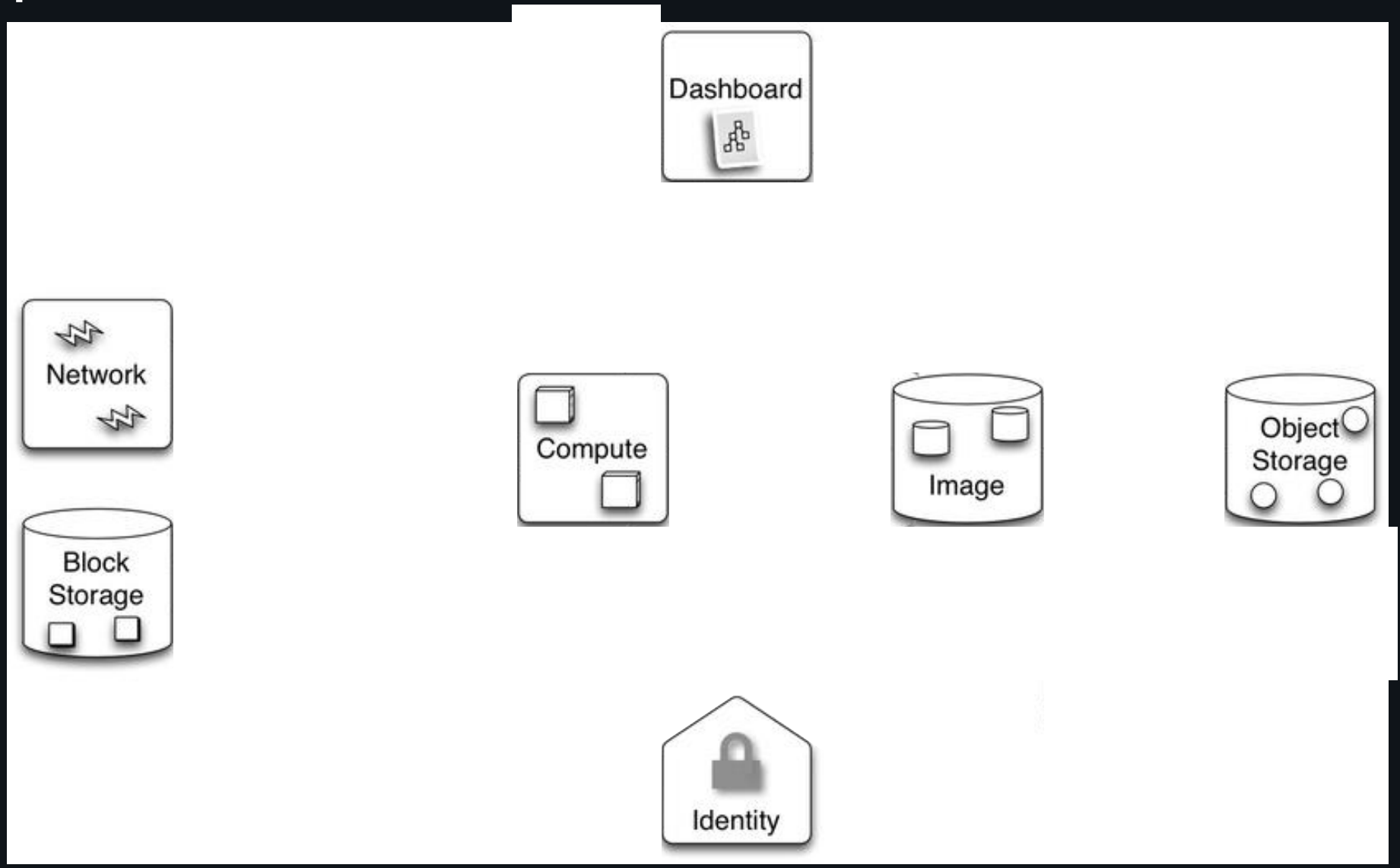


Details about OpenStack

- Allows for building a private cloud
- Common use cases
  - Public Cloud Offering (e.g. T-Systems Germany Open Telekom Cloud)
  - Private Cloud with multi tenancy (e.g. BMW runs environments for different business units)
  - Private Cloud (e.g. Prep Sportswear runs everything in one environment)
- Target: IaaS – running workloads in a virtualized environment
- Requirements:
  - Computing
  - Networking
  - Storage
  - Images

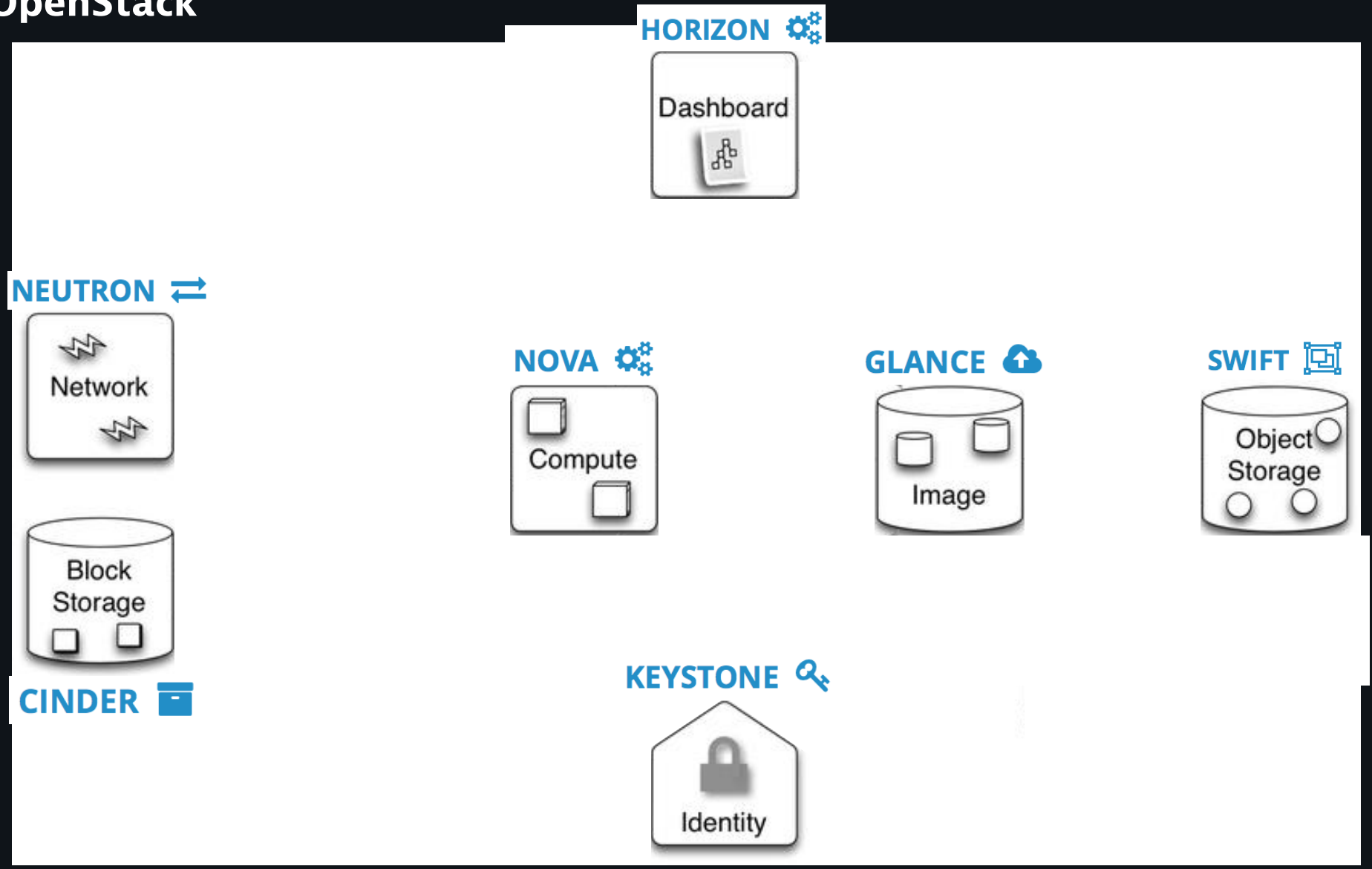
Details about OpenStack

- Micro service application
- Several core components
- Python



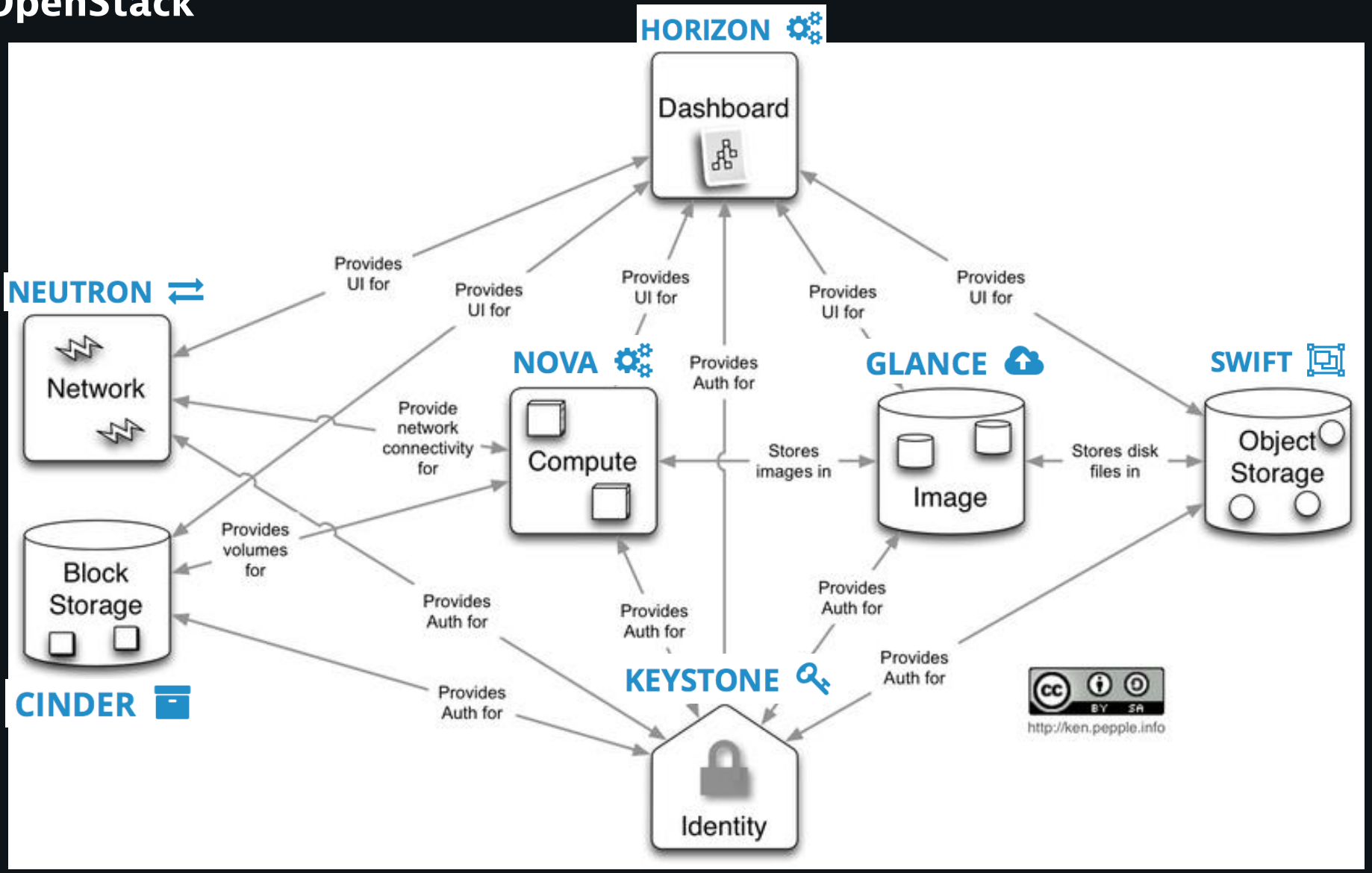
Details about OpenStack

- Micro service application
- Several core components
- Python



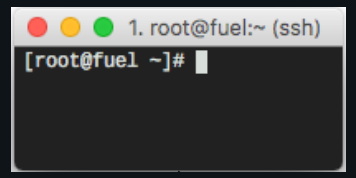
Details about OpenStack

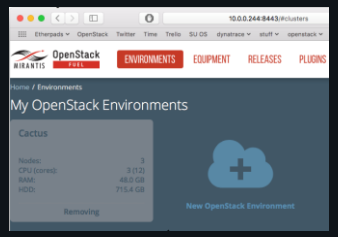
- Micro service application
- Several core components
- Python

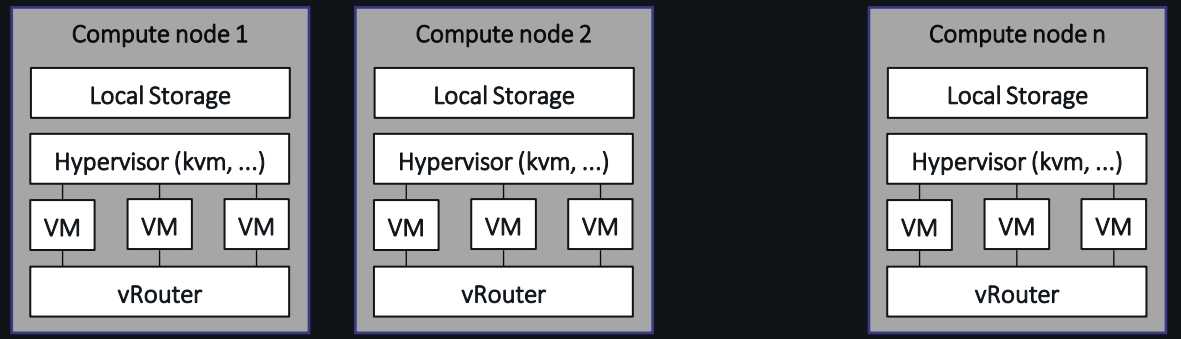


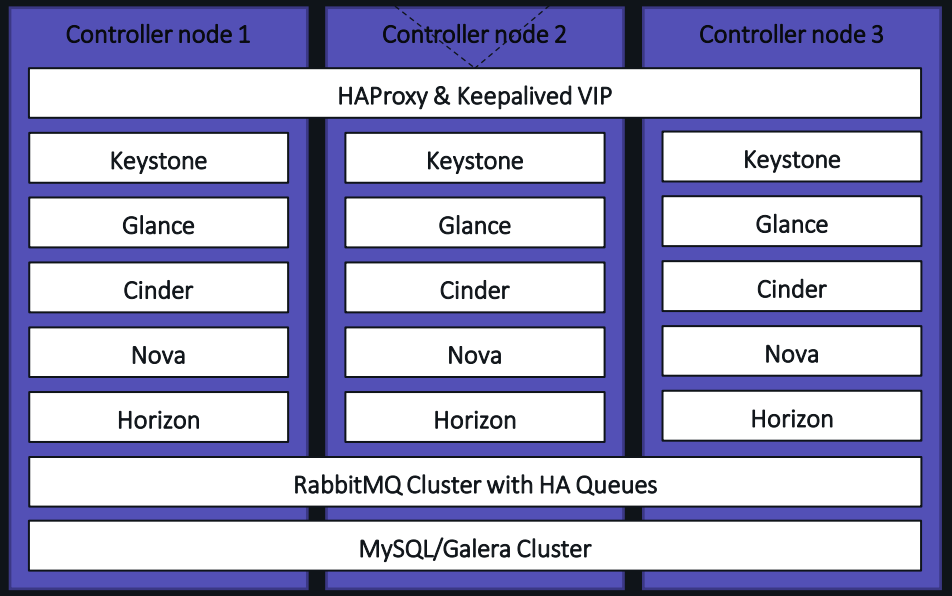
Details about OpenStack

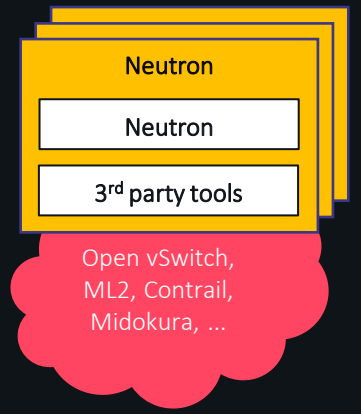
- Typical OpenStack cluster

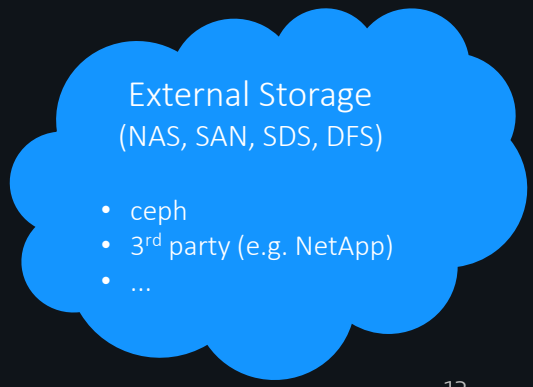






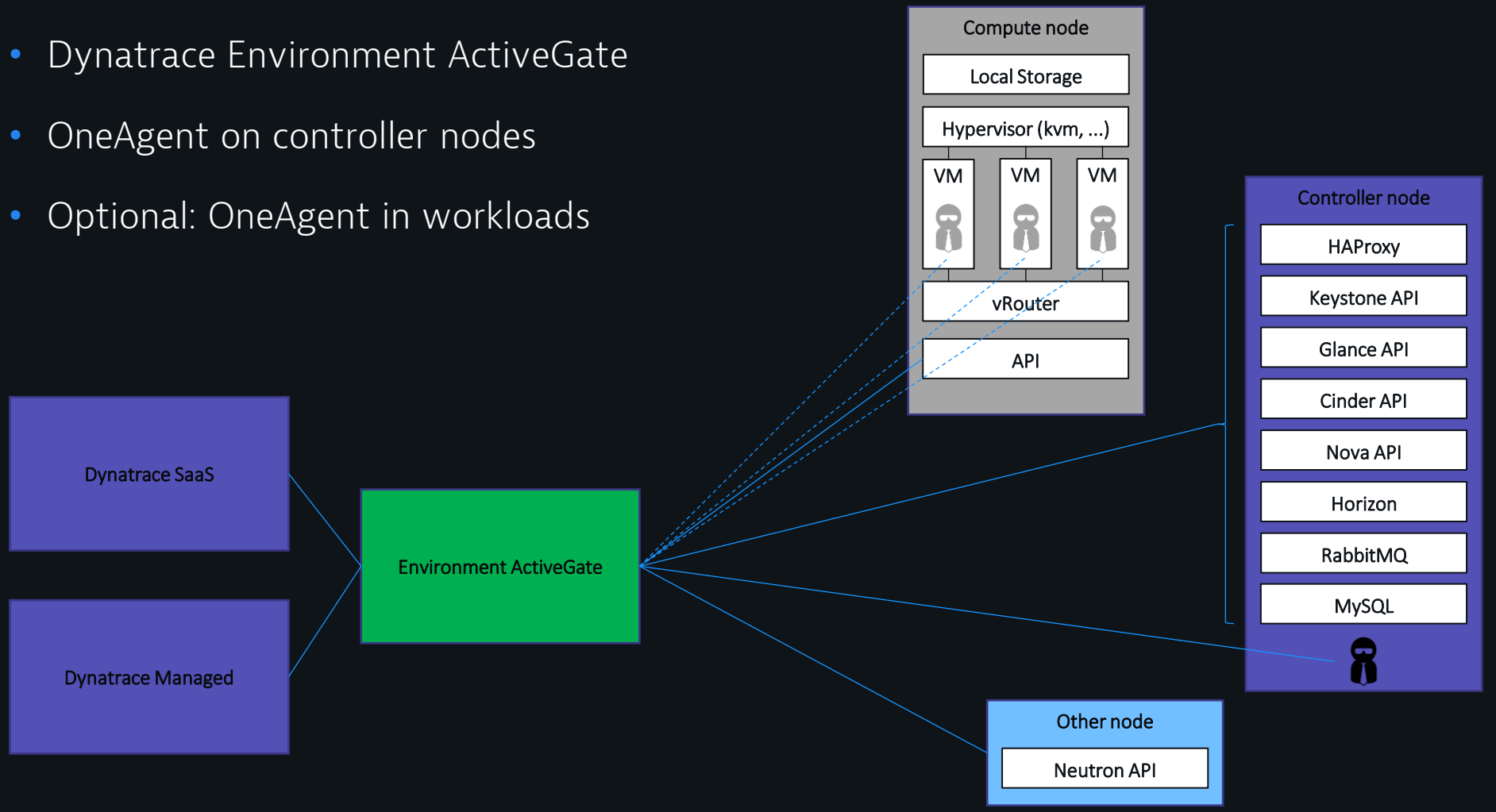






What and how do we monitor OpenStack

- Dynatrace Environment ActiveGate
- OneAgent on controller nodes
- Optional: OneAgent in workloads



Resources about OpenStack

- Official documentation: http://docs.openstack.org
- Glossary: http://docs.openstack.org/user-guide/common/glossary.html
- Dynatrace landing page: https://www.dynatrace.com/technologies/cloud-and-microservices/openstack-monitoring/

Simply smarter clouds