

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

The screenshot shows the Dynatrace Settings interface with the following navigation path: Settings > Cloud and virtualization > Overview.

Settings sidebar:

- Monitoring (selected, expanded):
 - Setup and overview
 - Monitored technologies
 - Monitoring overview
 - Host naming
- Processes and containers (collapsed)
- Web & mobile monitoring (collapsed)
- Cloud and virtualization (selected, expanded):
 - Connect vCenter, Azure, Cloud Foundry, K...
- Overview (selected)

Cloud & virtualization section:

Configure API based monitoring for the most popular virtualization and cloud types like AWS, VMWare, Azure, Cloud Foundry, and Kubernetes.

Monitored technologies:

- AWS**: Connect Amazon Web Services
- VMware**: Integrate with VMware
- Azure**: Connect Microsoft Azure
- Cloud Foundry**: Integrate with Cloud Foundry
- Kubernetes**: Integrate with Kubernetes

AWS

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

 AWS

Use this page to connect Amazon account to Dynatrace for monitoring. For AWS instances, go to your Amazon Console and prepare access for Dynatrace using either key- or role-based authentication. For details, see [How do I start AWS monitoring?](#)

Amazon will charge about \$0.01 per 1,000 requests to CloudWatch API and include the cost in the bill for the AWS account you use with Dynatrace.

[+ Connect new instance](#)

Name this connection

Access Key ID

Secret access key

AWS partition
▼

AWS resources monitoring method
▼

Monitor AWS resources tagged with any of the following tags (up to 2 entries):

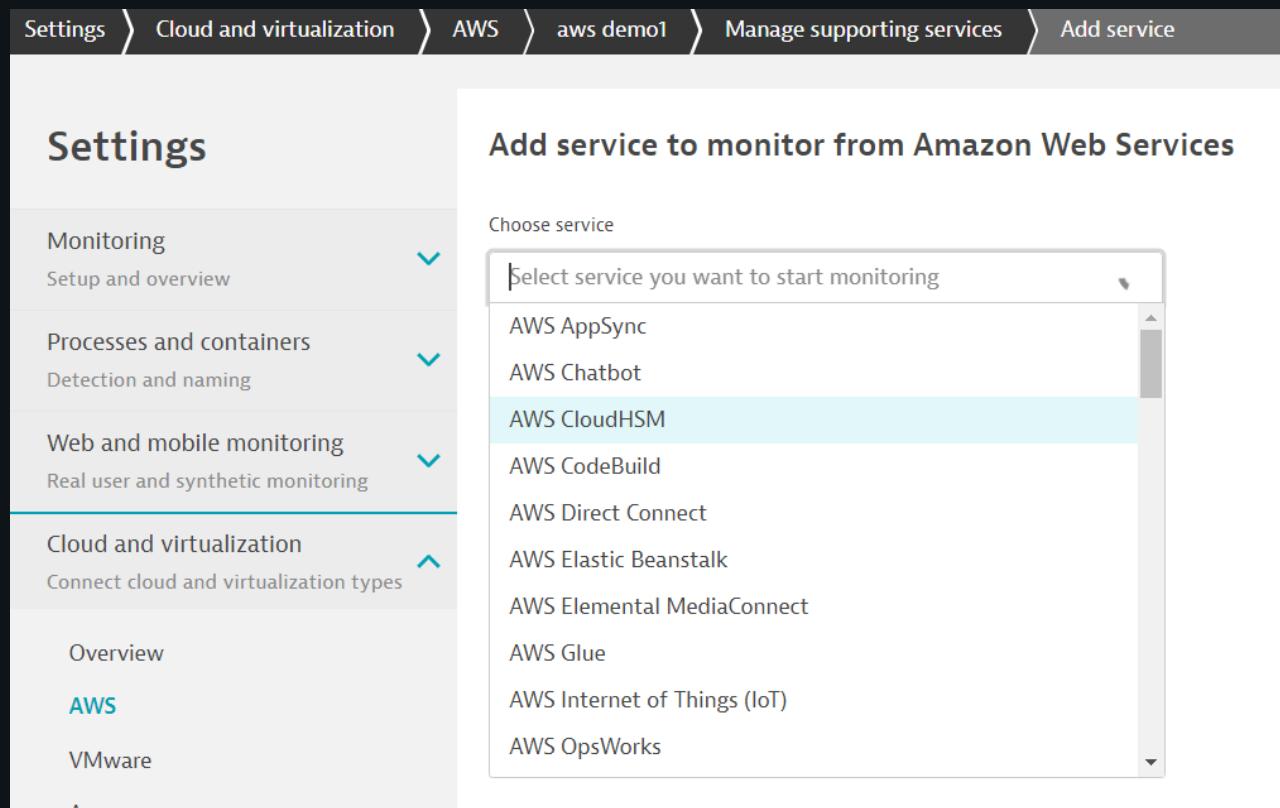
key	<input type="text" value="monitor_demo2"/>	value	<input type="text" value="true"/>
key	<input type="text"/>	value	<input type="text"/>

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

europe-west3-c - gke-idefix-default-pool-...
Uptime: 14 hours 53 minutes

Properties and tags

Kubernetes gcp-project: rnd-research k8s-worker + Add tag

Container-Optimized OS 69 (kernel 4.14.65+)

Detected name gke-idefix-default-pool-...
OneAgent version 1.155.236.20181024-141817
Architecture x86, 64-bit
Cloud Google Cloud Platform
Cloud platform type Kubernetes

GCE instance ID (highlighted by green box)
GCE instance name gke-idefix-default-pool-...
GCE machine type n1-standard-2
GCE project rnd-research
GCE project ID
GCE public IP addresses
GCE zone europe-west3-c (highlighted by green box)

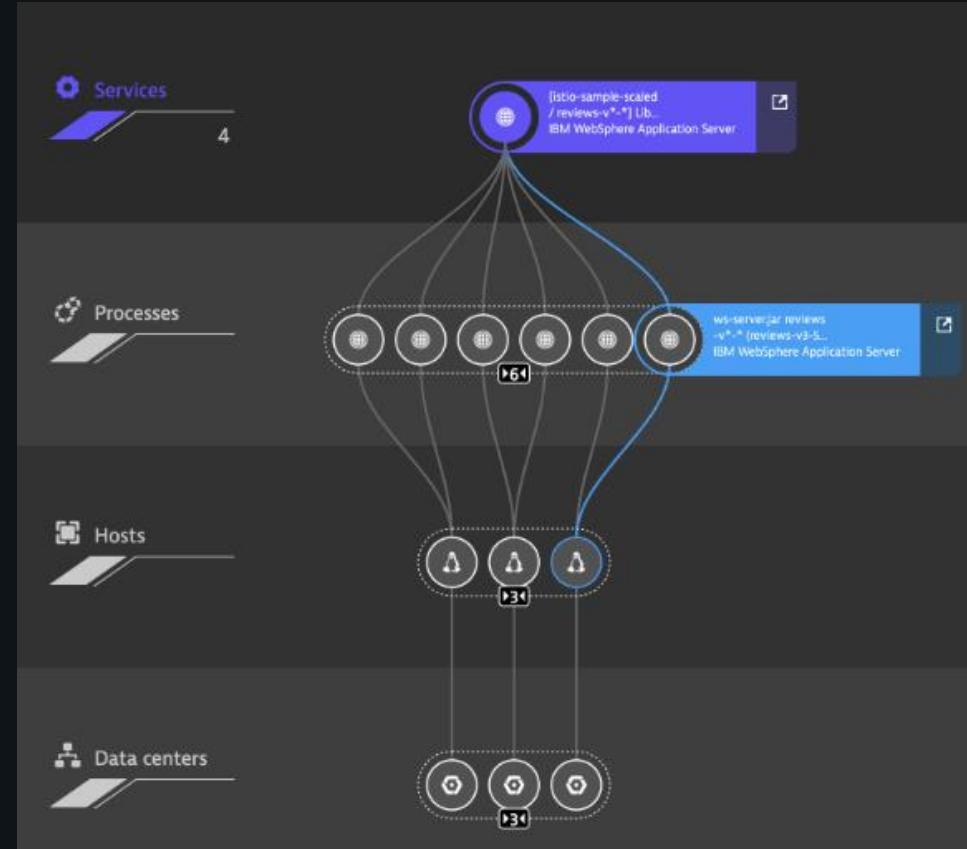
IP addresses 10.10.10.10
Logical CPU cores 2
Monitoring mode Full stack
Physical CPU cores 1
Virtualization KVM

11 % CPU
19 % Memory
1 NIC
3 Disks
1 Processes on 1 hosts

CPU usage 11 %

GCP in SmartScape

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



Demo



Azure environments

Infrastructure-as-a-Service

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

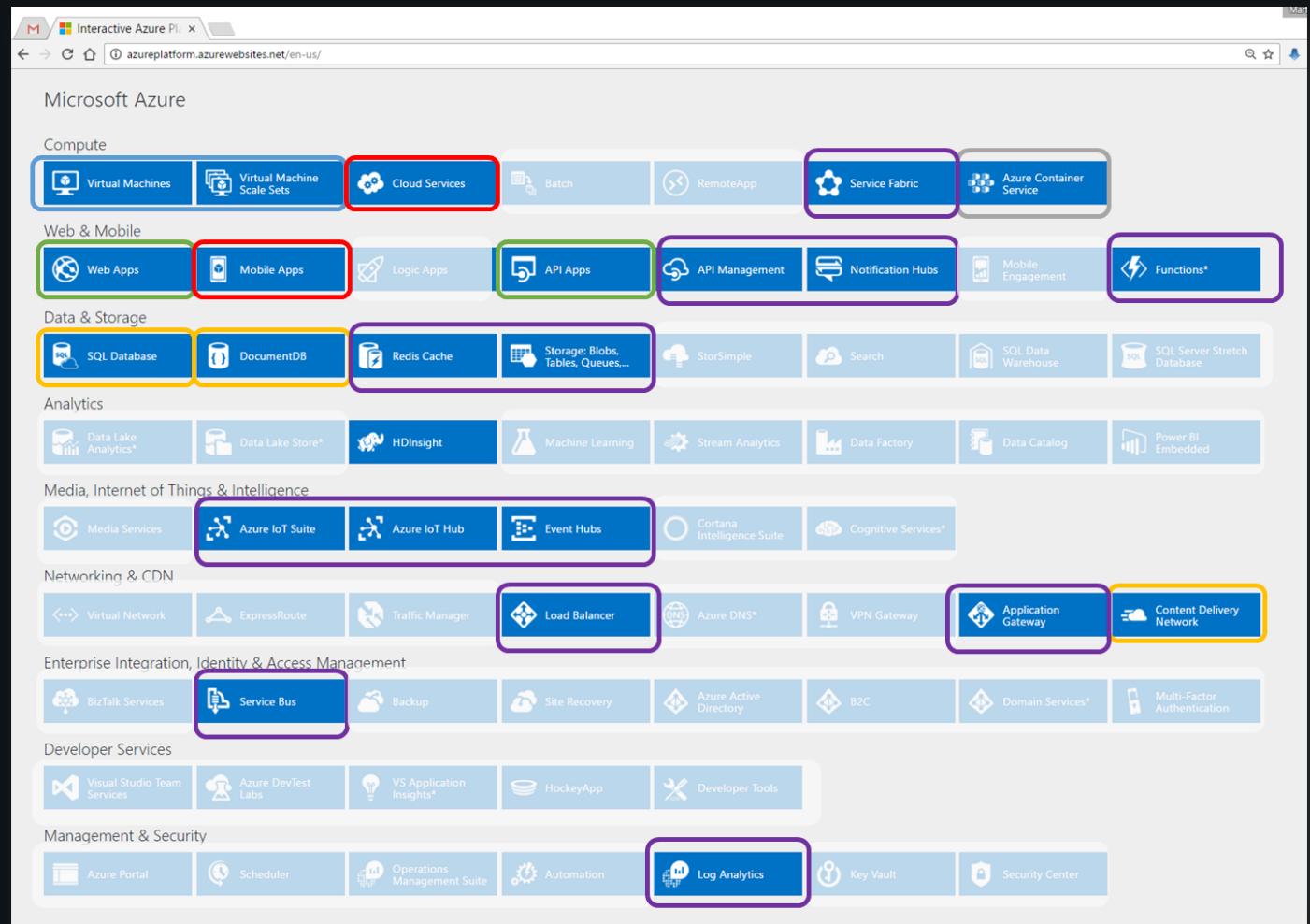
Service Bus

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

Microsoft Azure ▾

Dashboard ▾ + New dashboard Edit dashboard Share Fullscreen Clone Delete

Search resources x ! ⚙️ 😊 ?

All resources ALL SUBSCRIPTIONS

Service health MY RESOURCES

ruxittadoc NOSQL (DOCUMENTDB)

Resources VIKASRG2

vg160916f

Resources VGTESTRESOURCE

No resources in the resource group

DynaSiteCore

vg160916f

vg160916e

vg061016a

Marketplace

Help + support

DynaSiteCore

DynaSiteCore-ip

DynaSiteCore-nsg

See More

Resources TESTAUTOMATION

TestAutomationServic... APP SERVICE PLAN

ruxittadoc.net WEB APP

ruxittajava WEB APP

ruxittanode WEB APP

ruxittaphp WEB APP

ruxittapython WEB APP

ruxitTestAutomation... MYSQL DATABASE

Running

Running

Running

Running

Stopped

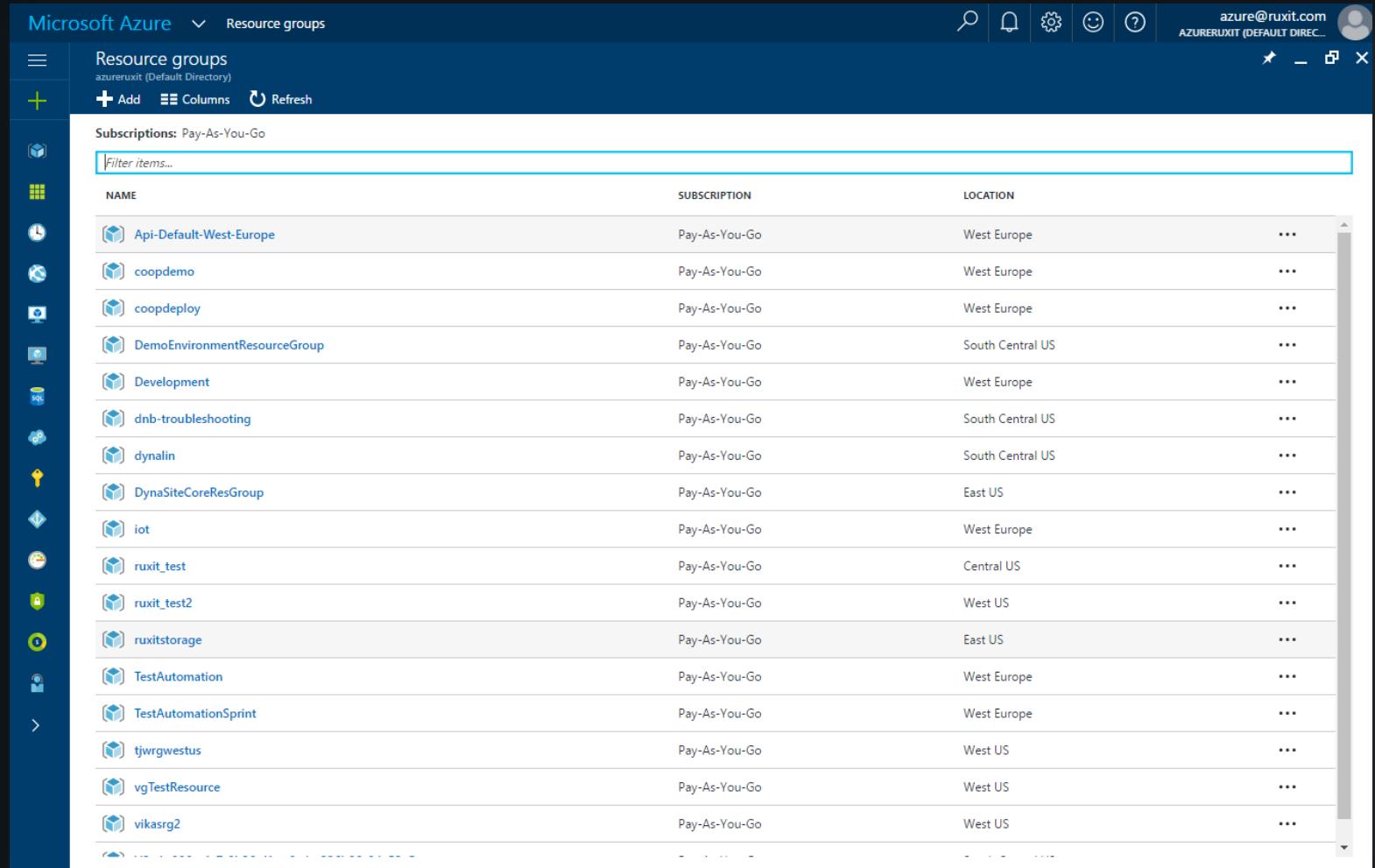
Healthy

See More

More services >

Resource Groups

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



The screenshot shows the Microsoft Azure Resource groups blade. The top navigation bar includes 'Microsoft Azure' (dropdown), 'Resource groups' (selected), a search icon, a bell icon, a gear icon, a smiley face icon, a question mark icon, and a user profile icon for 'azure@ruxit.com'.

The main area displays a table of resource groups:

NAME	SUBSCRIPTION	LOCATION	...
Api-Default-West-Europe	Pay-As-You-Go	West Europe	...
coopdemo	Pay-As-You-Go	West Europe	...
coopdeploy	Pay-As-You-Go	West Europe	...
DemoEnvironmentResourceGroup	Pay-As-You-Go	South Central US	...
Development	Pay-As-You-Go	West Europe	...
dnb-troubleshooting	Pay-As-You-Go	South Central US	...
dynalin	Pay-As-You-Go	South Central US	...
DynaSiteCoreResGroup	Pay-As-You-Go	East US	...
iot	Pay-As-You-Go	West Europe	...
ruxit_test	Pay-As-You-Go	Central US	...
ruxit_test2	Pay-As-You-Go	West US	...
ruxitstorage	Pay-As-You-Go	East US	...
TestAutomation	Pay-As-You-Go	West Europe	...
TestAutomationSprint	Pay-As-You-Go	West Europe	...
tjwrgwestus	Pay-As-You-Go	West US	...
vgTestResource	Pay-As-You-Go	West US	...
vikasrg2	Pay-As-You-Go	West US	...

Resource costs based on Resource Groups

The screenshot shows the Microsoft Azure portal interface. The left sidebar lists various resource groups under 'Subscriptions: Pay-As-You-Go'. The 'TestAutomationSprint' group is selected and highlighted in blue. The main content area displays the 'TestAutomationSprint - Resource costs' blade. This blade includes a navigation menu on the left with options like Overview, Activity log, Access control (IAM), Tags, Quickstart, Resource costs (which is selected and highlighted in blue), Deployments, Properties, Locks, Automation script, Metrics, Alert rules, Diagnostics logs, Application insights, and Log analytics (OMS). The right side of the blade shows a table titled 'Resource costs' with the following data:

NAME	TYPE	SPEND (USD)
TestAutoSprintServicePlan	App Service plan	208,80
ruxitSprintDotNet	App Service	0,34
ruxitSprintNode	App Service	0,27
ruxitSprintPhp	App Service	0,19
ruxitSprintJava	App Service	0,14
ruxitsprintdotnet	Application Insights	0,00
ruxitsprintjava	Application Insights	0,00
ruxitsprintnode	Application Insights	0,00
ruxitsprintphp	Application Insights	0,00

Extensions

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

The screenshot shows the Microsoft Azure portal interface. The left sidebar has a dark blue theme with various icons for different services. The main area shows a 'Virtual machines' blade for a 'DynaSiteCore' VM. A 'New resource' blade is open, specifically for 'DynaSiteCore - Extensions'. This blade lists several available extensions:

NAME	TYPE	STATUS
Acronis Backup (preview)		
Agent for Windows Server Monitoring		
APM Insight .NET Agent		
Control-M Agent		
Custom Script Extension		
Datadog Agent for Windows		
Deep Security Agent		
Dynatrace OneAgent		
ESET File Security		

The 'Extensions' section in the main blade shows a note: "Some details about the installed extensions are unavailable. This can occur when the virtual machine is stopped or the agent is unresponsive." Below this, a search bar and a table with columns 'NAME', 'TYPE', and 'STATUS' show "No resource extensions found."

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

¹Traces HTTP calls via HttpClient support

²Trace database calls via supported database frameworks (for example, ADO.NET or JDBC).

³Only available for Standard Load Balancer

⁴End-to-End tracing via Trace Context

⁵End-to-End tracing using OneAgent SDK

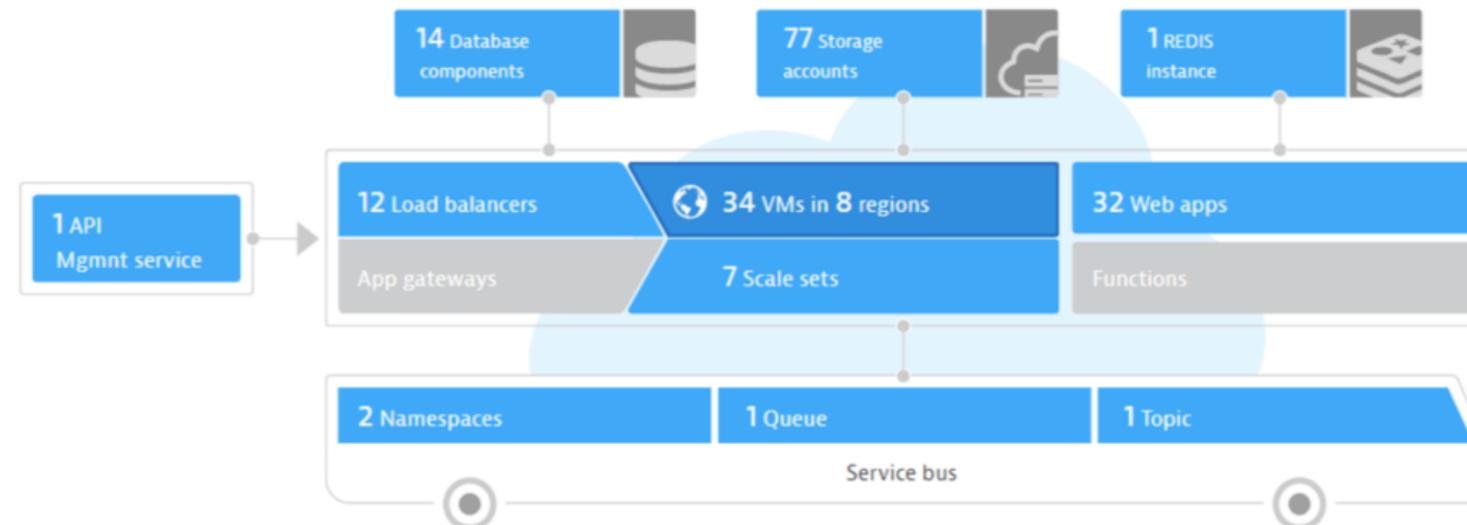
Deploying applications w/ Dynatrace

Deploying Monitoring for Azure

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

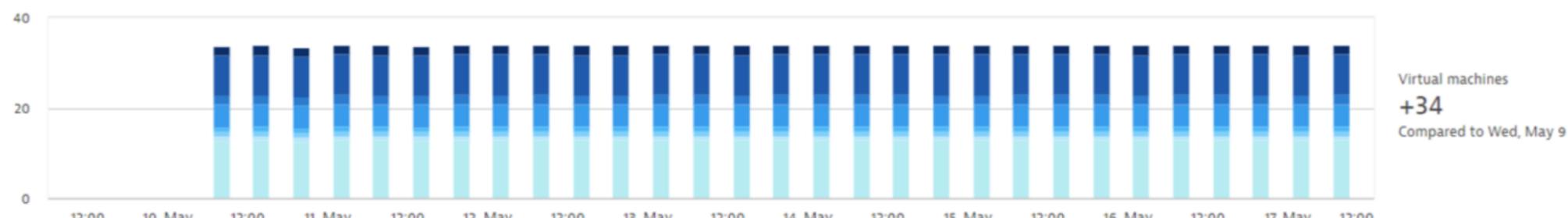
Enhanced Azure Service Insights

The screenshot shows the Dynatrace Settings interface. The navigation bar at the top includes 'Settings' > 'Cloud and virtualization' > 'Azure'. The main area is titled 'Settings' and contains a sidebar with sections: 'Monitoring' (selected), 'Monitoring overview', 'Monitored technologies', 'Process groups', 'Detection and naming', 'Web & mobile monitoring' (selected), 'Real user & synthetic monitoring', 'Cloud and virtualization' (selected), 'Overview', 'AWS', 'VMware', 'OpenStack', and 'Azure' (highlighted with a green arrow). The main content area is titled 'Azure' and provides instructions for connecting to Azure for monitoring. It states: 'Use this page to connect Azure account to Dynatrace for monitoring. Sign in to Azure Management Portal and prepare access to Dynatrace using either Client ID or Tenant ID and the Secret Key. For details, see [How do I start Azure monitoring?](#)' and a note about potential charges. Below this are four input fields: 'Name this connection' (with example 'E.g. Dynatrace integration'), 'Subscription Name' (with example 'E.g. 98989ae2-4566-4efd-9db8-e194bb3910e4'), 'Client ID' (with example 'E.g. 68345fd1-3216-4aed-7be2-a244bb3907b2'), 'Tenant ID' (with example 'E.g. 68345fd1-3216-4aed-7be2-a244bb3907b2'), 'Secret Key' (with example 'Key's Value'), and two buttons at the bottom: 'Connect' and 'Cancel'. A large green arrow points from the 'Azure' link in the sidebar to the 'Secret Key' field.



Environment dynamics

Average number of virtual machines over last 7 days



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

The screenshot shows the Dynatrace interface for PaaS integration. At the top, there's a search bar and navigation links for 'Deploy Dynatrace' and 'PaaS Integration'. The main section is titled 'PaaS integration' and features two cards: one for 'Azure Web Apps' and one for 'Cloud Foundry'. Both cards include a green circular icon, a title, a link to 'How do I integrate [platform] apps?', and a 'Copy' button next to an input field containing an environment ID or token. Below these cards, a note says 'You'll need your environment ID and an API token to complete setup at your PaaS platform portal.' There are also 'Environment ID' and 'API token' fields with their respective values displayed.

PaaS integration

Azure Web Apps
How do I integrate Azure Web Apps?

Cloud Foundry
How do I integrate Cloud Foundry apps?

You'll need your environment ID and an API token to complete setup at your PaaS platform portal.

Environment ID

kwl61035

API 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

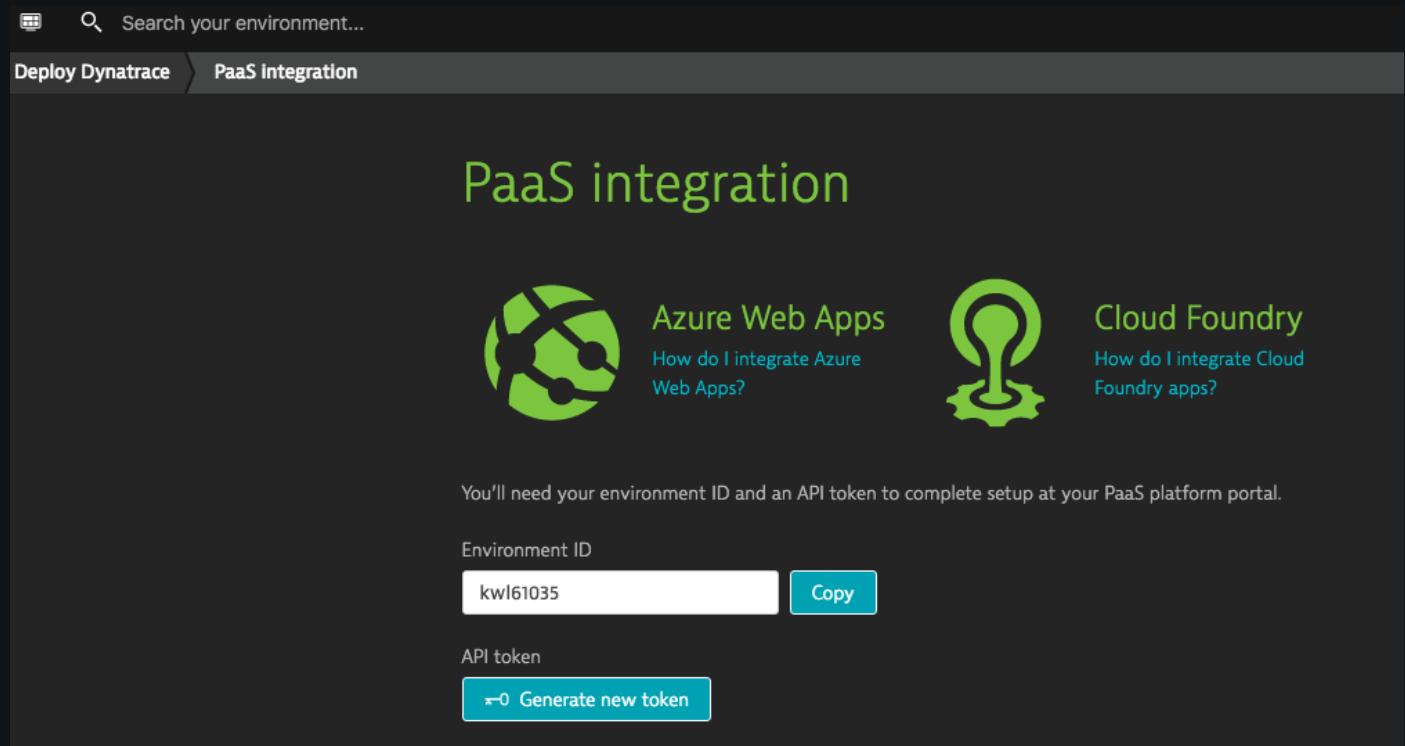
The screenshot shows the Dynatrace Monitoring overview page. On the left, a sidebar menu is open under the 'Monitoring' section, with 'Monitoring overview' selected. The main area is titled 'Monitoring overview' and contains instructions to enable monitoring for hosts, process groups, and applications. It features four buttons: 'Monitor another host', 'Install Security Gateway', 'Set up agentless or AMP monitoring', and 'Monitor a mobile app'. Below these are links to download OneAgent or Security Gateway installers and to customize process detection. The central part of the page displays three tabs: 'Hosts (5)', 'Process groups (21)', and 'Applications (0)'. The 'Process groups (21)' tab is active. A table lists four process groups, all of which have 'Service monitoring: Off / On' status set to 'Off'. Each row includes a 'Summary' column and an 'Edit' button. The rows are:

Process group	Summary	Service monitoring: Off / On	Edit
.NET HealthMetrics.DoctorService.exe	2 services, 1 instance waiting for restart	Off	Edit
.NET HealthMetrics.BandService.exe	1 service, 1 instance waiting for restart	Off	Edit
.NET HealthMetrics.NationalService.exe	1 service, 1 instance waiting for restart	Off	Edit
.NET HealthMetrics.WebService.exe	1 service, 1 instance waiting for restart	Off	Edit

Monitor Azure Web Apps

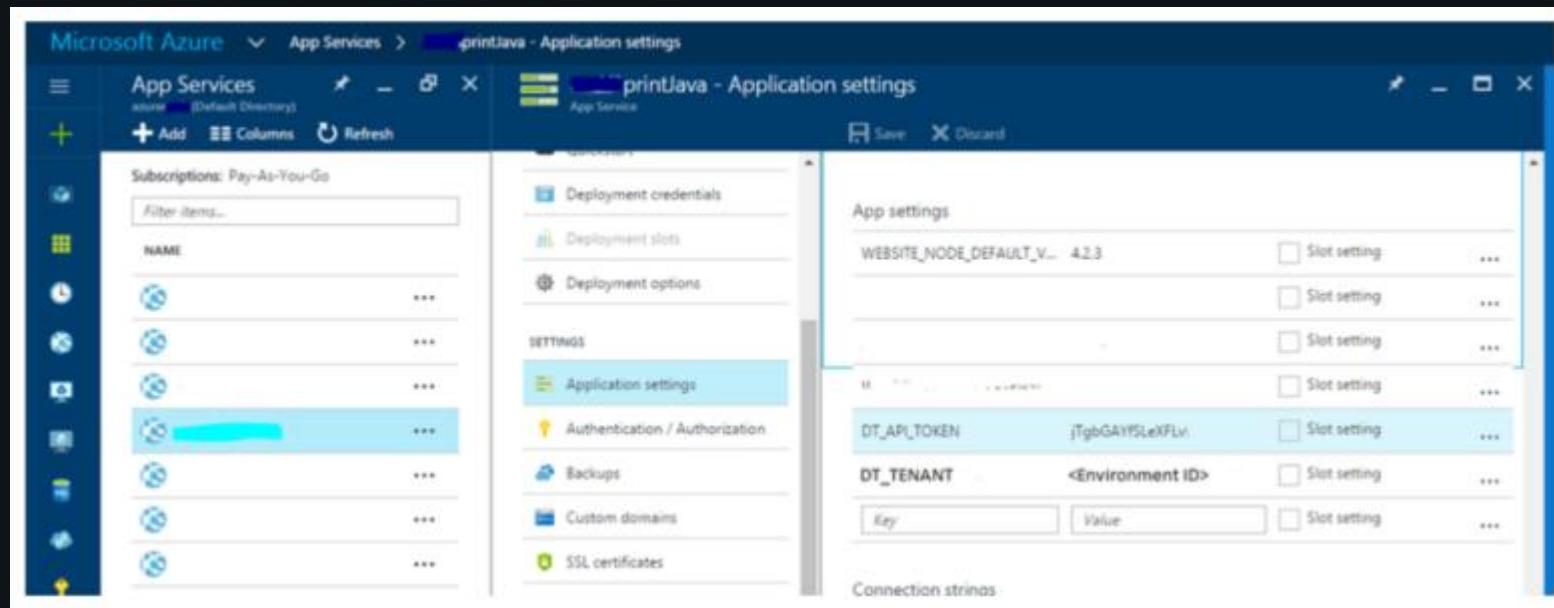
Monitor a Web App

- Step 1
 - Generate PaaS Token



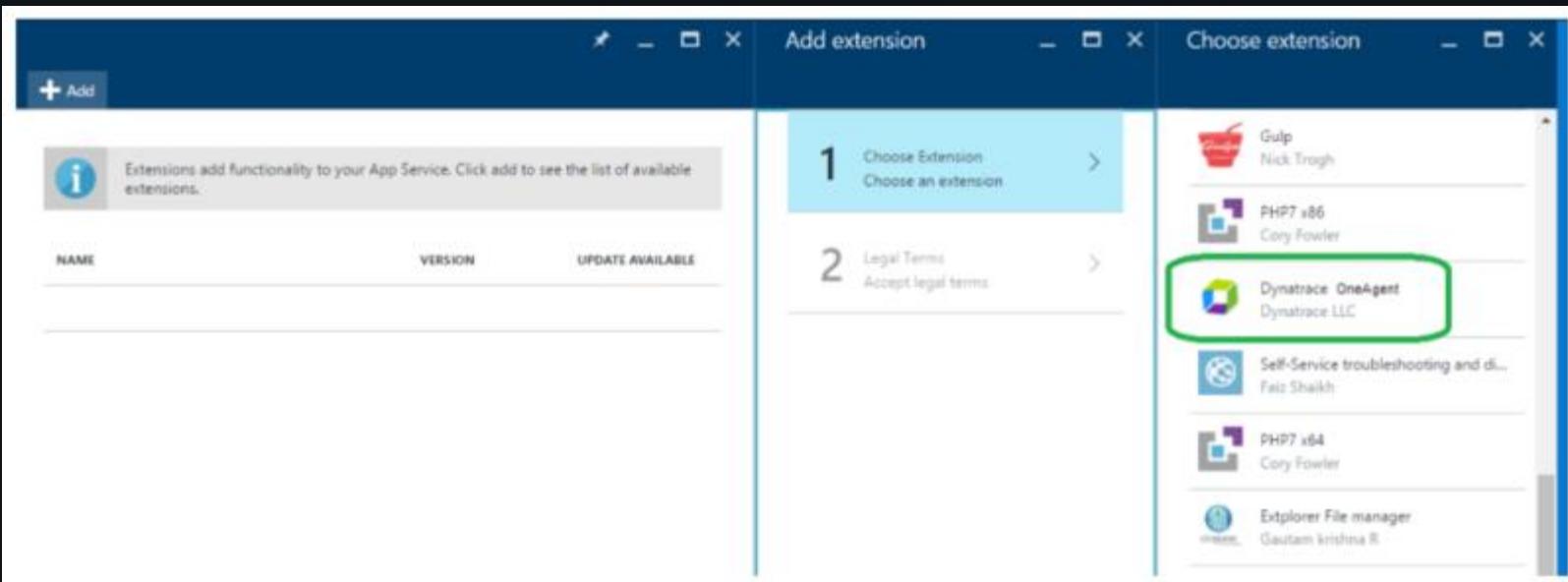
Monitor a Web App

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



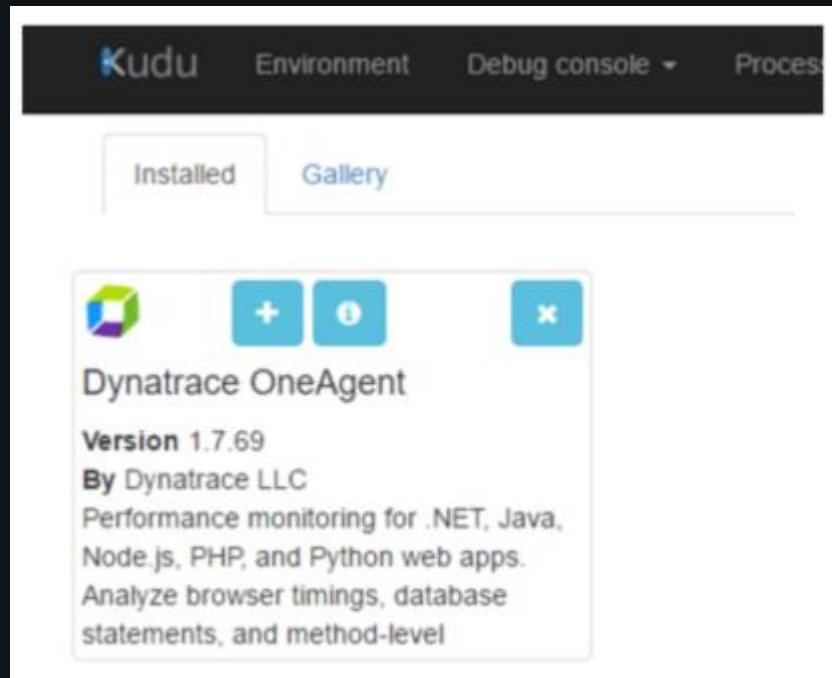
Monitor a Web App

- 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

The screenshot shows the Dynatrace interface for PaaS integration. At the top, there's a search bar and a navigation bar with 'Deploy Dynatrace' and 'PaaS Integration'. The main title is 'PaaS integration'. Below it, there are two sections: 'Azure Web Apps' and 'Cloud Foundry'. Each section has a green icon, a title, and a link to 'How do I integrate [platform] apps?'. A note at the bottom says 'You'll need your environment ID and an API token to complete setup at your PaaS platform portal.' It shows an 'Environment ID' input field containing 'kwl61035' with a 'Copy' button, and an 'API token' input field with a 'Generate new token' button.

PaaS integration

Azure Web Apps
How do I integrate Azure Web Apps?

Cloud Foundry
How do I integrate Cloud Foundry apps?

You'll need your environment ID and an API token to complete setup at your PaaS platform portal.

Environment ID
kwl61035

API token

Monitor Compute VMs

- Install the Dynatrace OneAgent Azure VM Extension via Azure Portal

The screenshot shows the Azure portal interface for managing extensions on a virtual machine named "winarmext". The left sidebar contains navigation links: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, SETTINGS, Availability set, Disks, Extensions, and Network interfaces. The "Extensions" link is highlighted with a blue background. The main content area has a header "winarmext - Extensions" and a sub-header "Virtual machine". It features a search bar labeled "Search (Ctrl+ /)" and a button labeled "+ Add". Below these is a table with columns "NAME" and "TYPE". A message at the top of the table says "No resource extensions found."

Monitor Compute VMs

- Configure the OneAgent Extension

The screenshot shows the Microsoft Azure portal interface for installing a new resource. The top navigation bar includes 'Microsoft Azure', 'Virtual machines', 'winarmtest - Extensions', 'New resource', 'Dynatrace OneAgent', and 'Install extension'. A search bar labeled 'Search resources' is also present.

The main area displays a list of available extensions on the left, with the 'Dynatrace OneAgent' extension selected and highlighted in blue. The selected extension's details are shown in the center:

- Description:** Dynatrace provides full-stack application performance monitoring using advanced Artificial Intelligence technology. It helps DevOps teams to faster resolve problems and support operational decisions. The Dynatrace OneAgent is easy to install and provides immediate information on applications, infrastructure and service problems. Dynatrace analyzes how application services interact and visualizes the impact on overall application performance.
- Publisher:** Dynatrace
- Useful Links:**
 - Dynatrace Azure - Overview
 - Dynatrace - Help
 - Dynatrace corporate website
 - End User License Agreement

On the right side, the 'Install extension' form is displayed, requiring two inputs:

- * Environment ID (input field)
- * API Token (input field)

Demo

Enterprise Cloud Platforms

Cloud Foundry

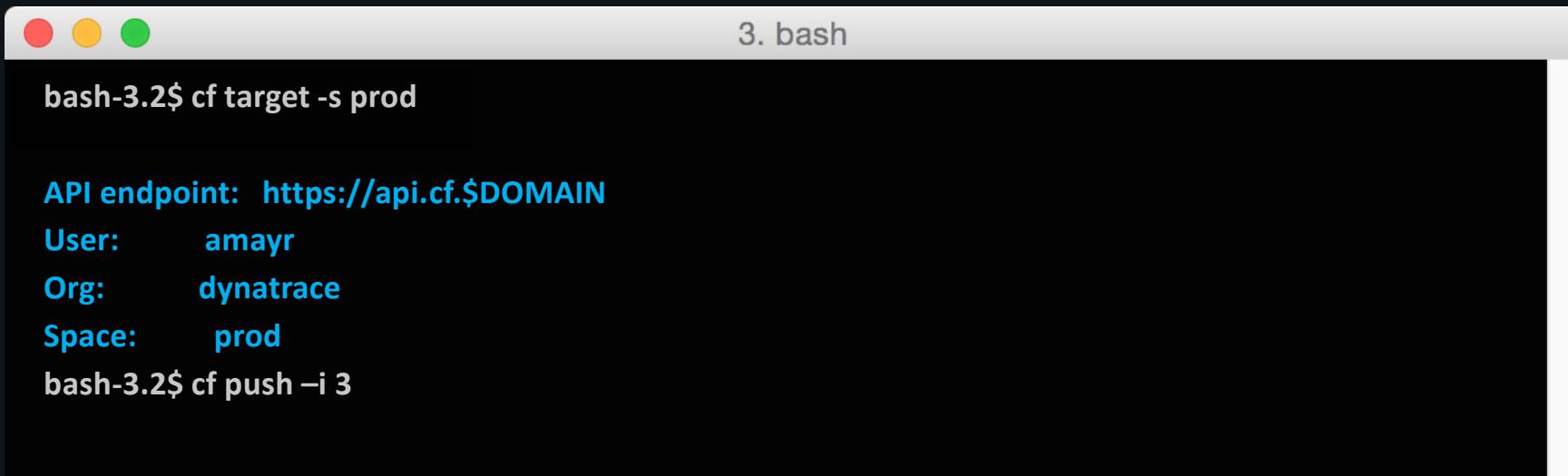


What is Cloud Foundry?

- 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,...



The screenshot shows a terminal window with a dark background and light-colored text. At the top left are three small colored circles (red, yellow, green). The title bar reads "3. bash". The terminal content includes:

```
bash-3.2$ cf target -s prod

API endpoint: https://api.cf.$DOMAIN
User: amayr
Org: dynatrace
Space: prod
bash-3.2$ cf push -i 3
```



Cloud Delivery Models

Traditional on Premise

Applications
Data
Runtime
Middleware
OS
Virtualization
Servers
Storage
Networking

Infrastructure-as-a-Service

Applications
Data
Runtime
Middleware
OS
Virtualization
Servers
Storage
Networking

Platform-as-a-Service

Applications
Data
Runtime
Middleware
OS
Virtualization
Servers
Storage
Networking

Software-as-a-Service

Applications
Data
Runtime
Middleware
OS
Virtualization
Servers
Storage
Networking

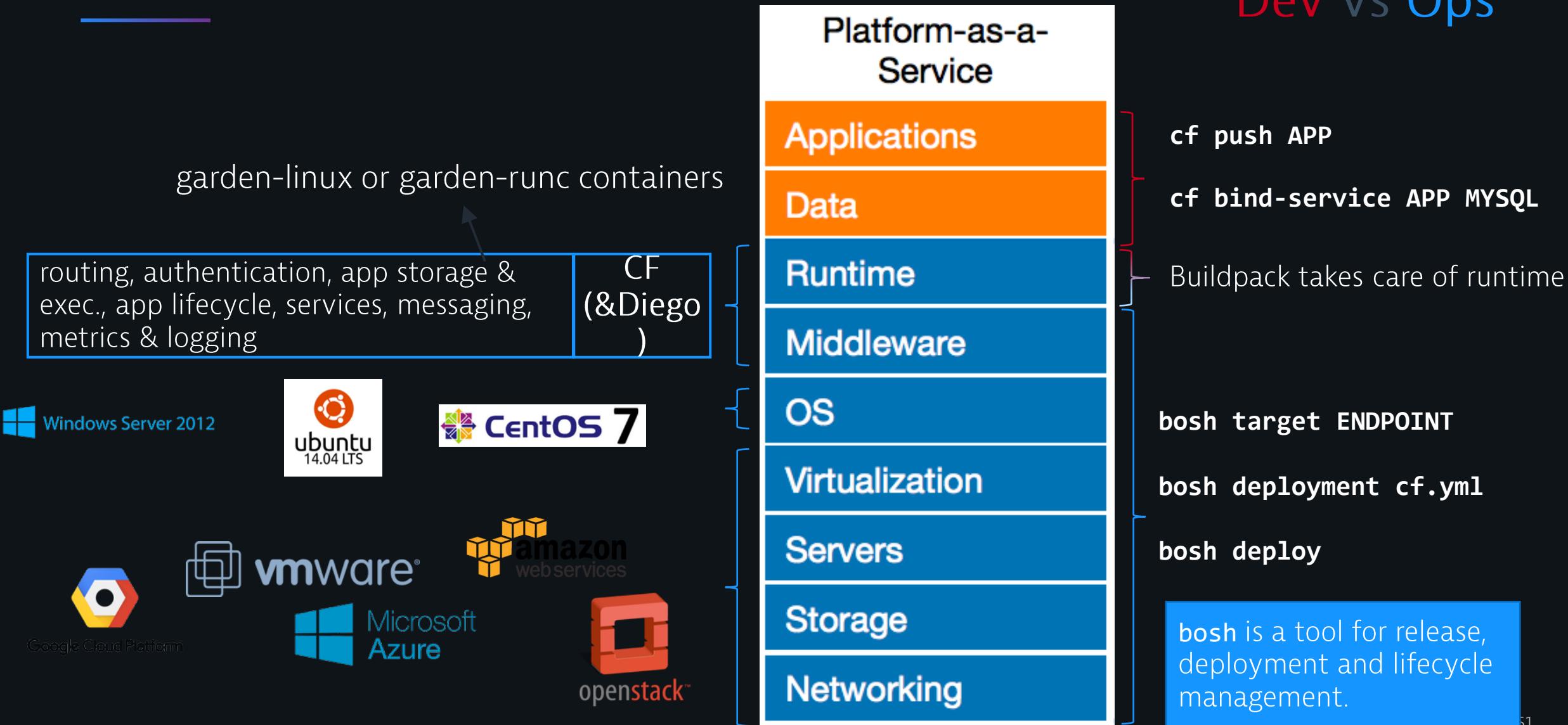
Client Manages

Vendor Manages in Cloud

Standardization – Lower Costs –Faster Time to Value

The Cloud Foundry Stack

Dev vs Ops



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)



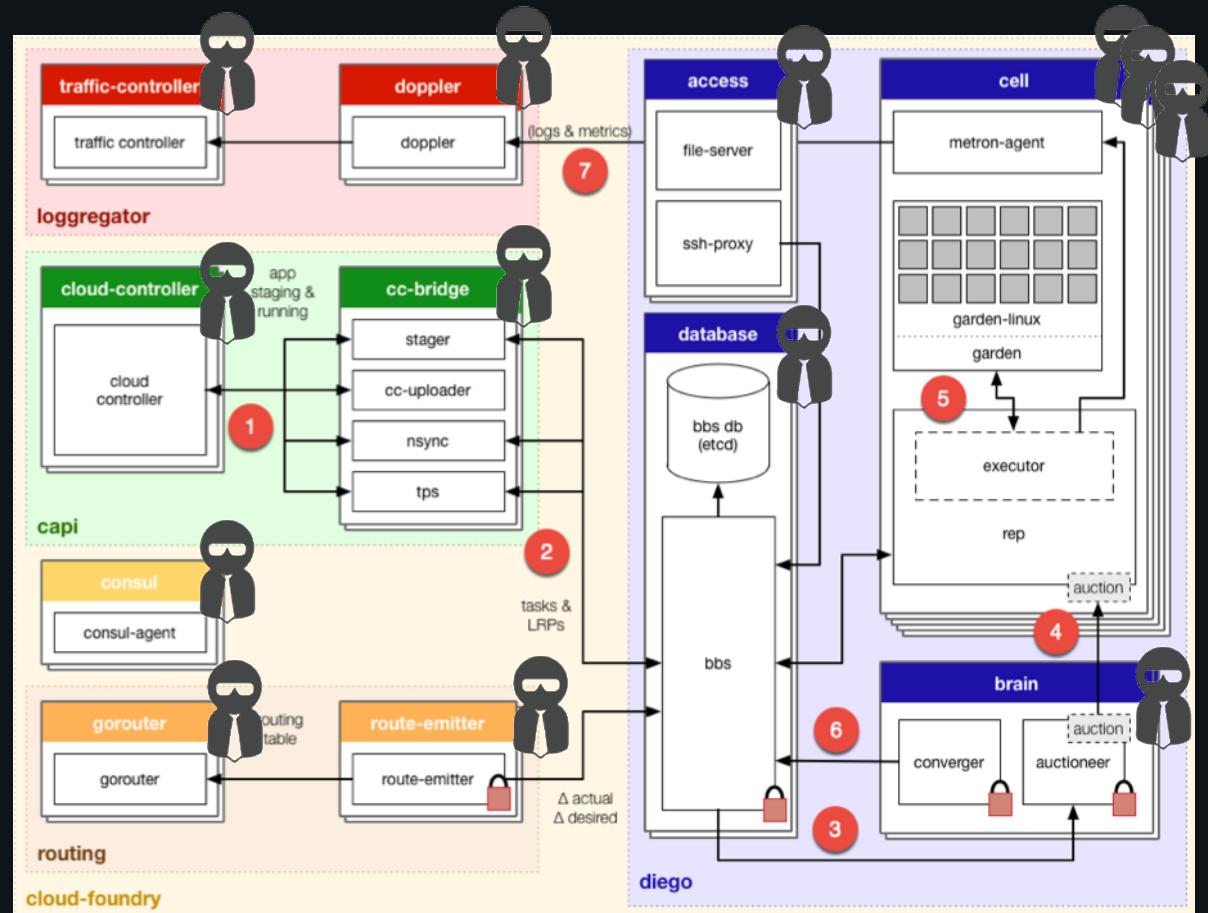
Enable Dynatrace via CF service

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

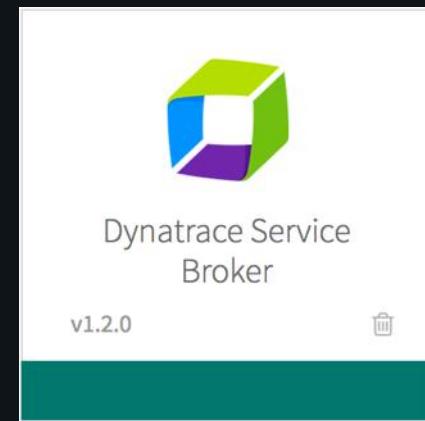
- Deploying full OneAgent to each CF related VM
 - Via bosh-addon
- Auto-injection into garden containers (similar to Dynatrace Docker support)
- No buildpack integrations or additional services needed
- Complete CF cluster in one Dynatrace environment
- Requirements:
 - CF architecture already based on garden-runc containers (present in CF releases running on stemcells based on 4.4 kernel)
 - Full access to host



Flavors of Cloud Foundry

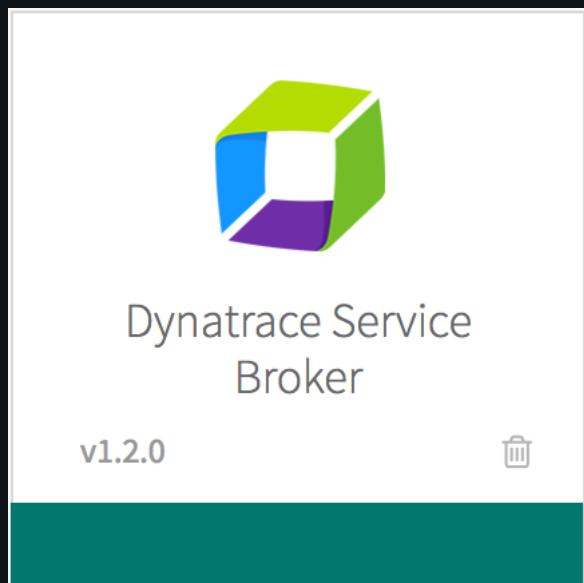
Pivotal Cloud Foundry

- 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



[Installation Dashboard](#)

Dynatrace Service Broker

Settings Status Credentials Logs

Assign AZs and Networks

SaaS/Managed Plans

AppMon Plans

Service Access

Errands

Resource Config

Stemcell

Operator-Defined Service Plans for Dynatrace SaaS/Managed

Dynatrace SaaS/Managed Service Plans

kwl

example-name

Plan Name *

example-name

Environment ID *

someenvironmentid

API token *

somepitoken

API token for accessing Dynatrace OneAgent download.

API base URL

Save

PCF Ops Manager v1.8.7.0; ©2013-2016 Pivotal Software, Inc; All Rights Reserved.

API Docs | End User License Agreement



Other Distributions...



PREDIX

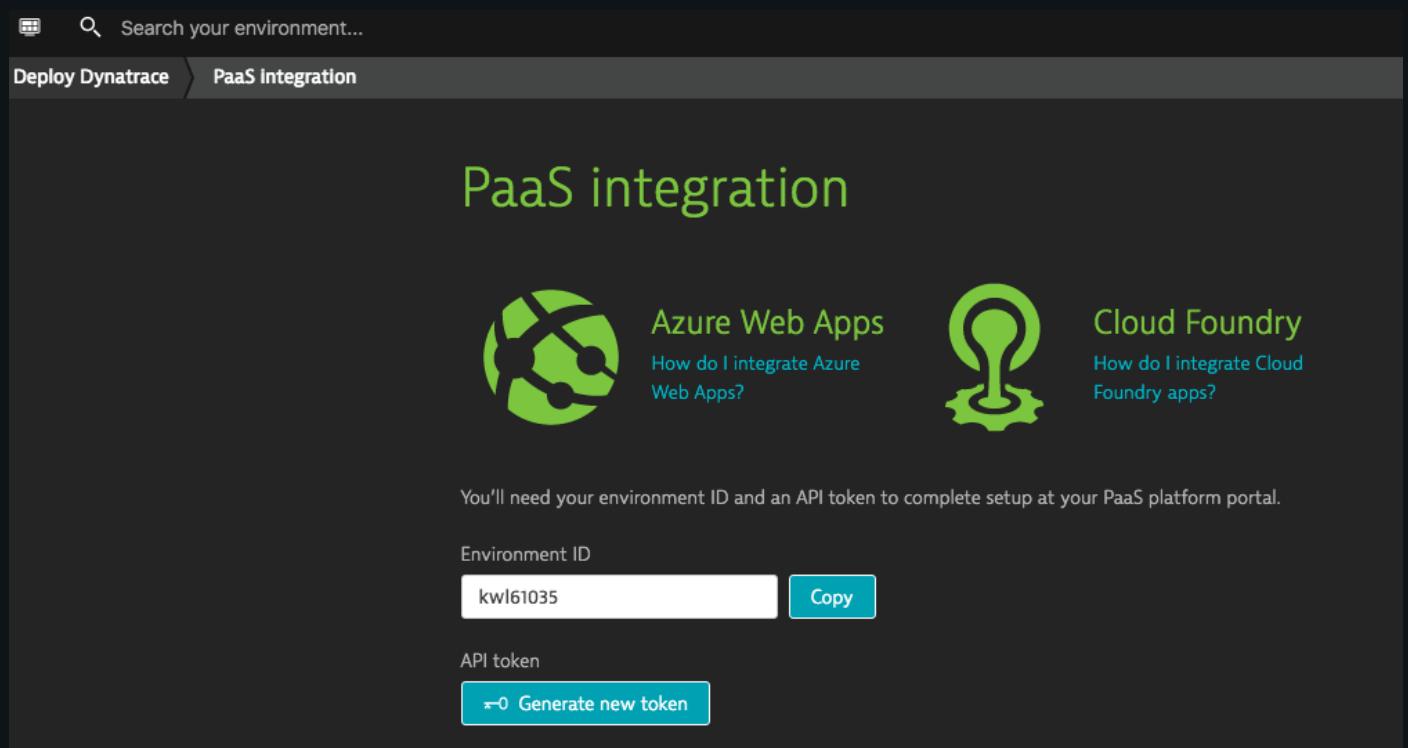


Atos

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
 - 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.

```
---  
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
```

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

The screenshot shows the Dynatrace interface for PaaS integration. At the top, there's a search bar and navigation links for 'Deploy Dynatrace' and 'PaaS Integration'. The main section is titled 'PaaS integration' and features two main items: 'Azure Web Apps' and 'Cloud Foundry'. Each item has a green circular icon, a title, and a link to 'How do I integrate [platform]?' Below these, a note says 'You'll need your environment ID and an API token to complete setup at your PaaS platform portal.' There are input fields for 'Environment ID' (containing 'kwI61035') and 'API token', with a 'Copy' button next to the environment ID and a 'Generate new token' button next to the API token.

PaaS integration

Azure Web Apps
How do I integrate Azure Web Apps?

Cloud Foundry
How do I integrate Cloud Foundry apps?

You'll need your environment ID and an API token to complete setup at your PaaS platform portal.

Environment ID
kwI61035 Copy

API token
→0 Generate new 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
- Preferred method for:
 - Google App Engine
 - Amazon ECS Fargate
 - Other locked-down K8s based platforms
- Several options available...

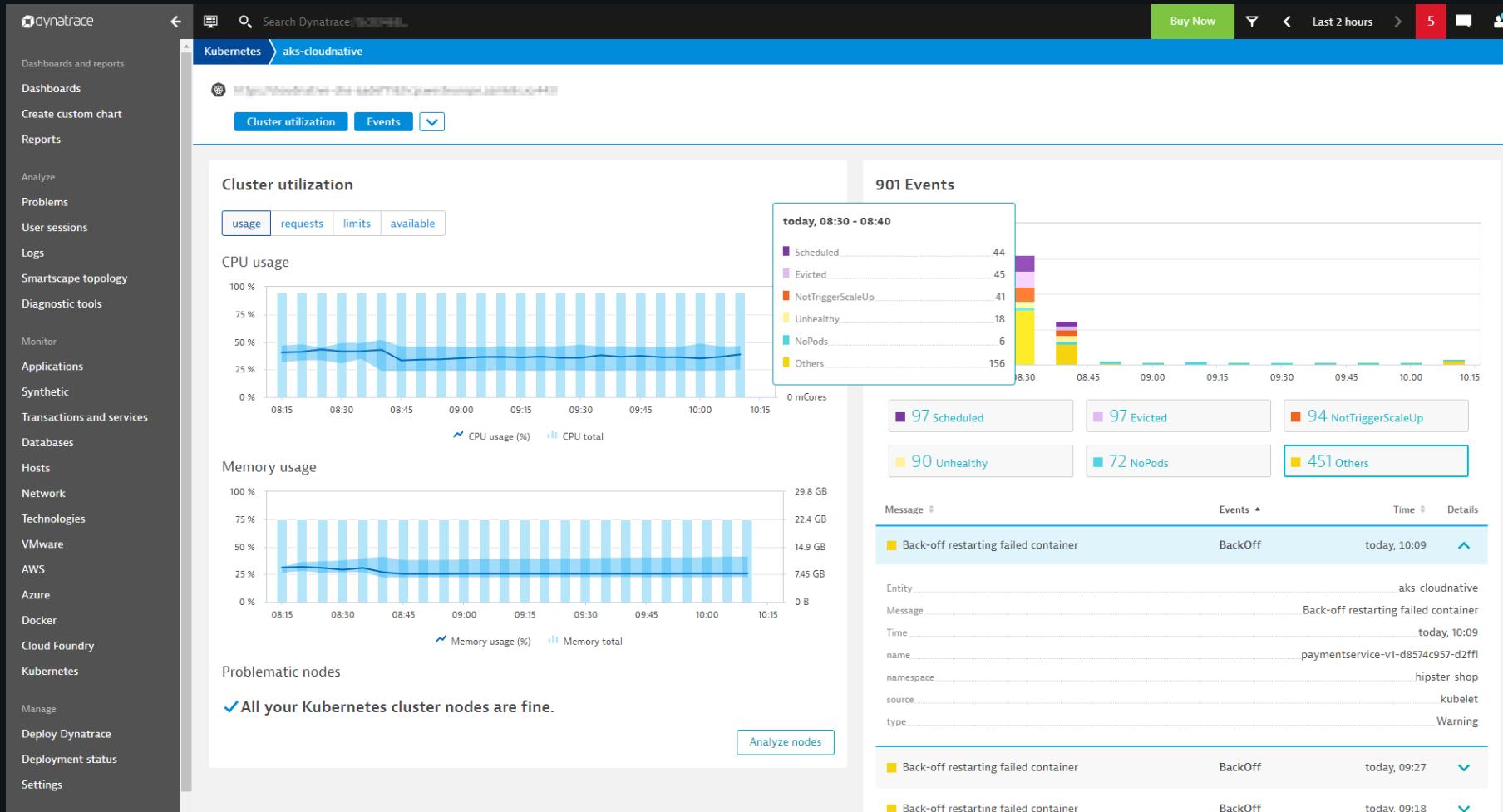
Kubernetes Application-Only Monitoring Strategies

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`.

Events Field Selectors

Define Kubernetes event filters to ingest events into your environment. For more details, see the [documentation](#).

[+ Add events field selector](#)

The screenshot shows a configuration dialog for a 'Field selector name' named 'hipster-shop all'. The 'Field selector expression' is set to 'involvedObject.namespace=hipster-shop'. The 'Active' toggle switch is turned on. At the bottom are 'Apply' and 'Cancel' buttons.

hipster-shop all true

Field selector name
hipster-shop all *

Field selector expression
involvedObject.namespace=hipster-shop *

Active: Off/On

Apply Cancel

Pod + Namespace Oriented Monitoring

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

The screenshot shows a monitoring interface titled "Cloud applications" with the subtitle "Overview of your cloud applications". At the top, there are three filter buttons: "Filtered by Current pod state: Pending" (with a red X), "Current pod state: Failed" (with a red X), and "Clear all". Below the filters, the title "6 Cloud applications" is displayed, followed by a table with the following data:

Name	Namespace	Deployment types	Running	Memory usage	CPU usage	CPU throttling
cartservice 1 Pod in Pending state	hipster-shop	Deployment	2 of 3	1.04 GB	13.2 %	28.097 s/min
fluentd 1 Pod in Pending state	dynatrace	DaemonSet	3 of 4	-	-	-
istio-galley 1 Pod in Failed state	istio-system	Deployment	1 of 1	3.59 GB	33.1 %	< 1 ms/min
istio-telemetry 2 Pods in Pending state	istio-system	Deployment	1 of 3	1.67 GB	103.8 %	< 1 ms/min
nginx-ingress-nginx-ingress 2 Pods in Failed state	default	Deployment	1 of 1	125 MB	0.2 %	< 1 ms/min
shippingservice 1 Pod in Pending state	hipster-shop	Deployment	1 of 2	66.2 MB	1 %	217.117 ms/min

At the bottom center of the table is a page navigation bar with arrows and the number "1".

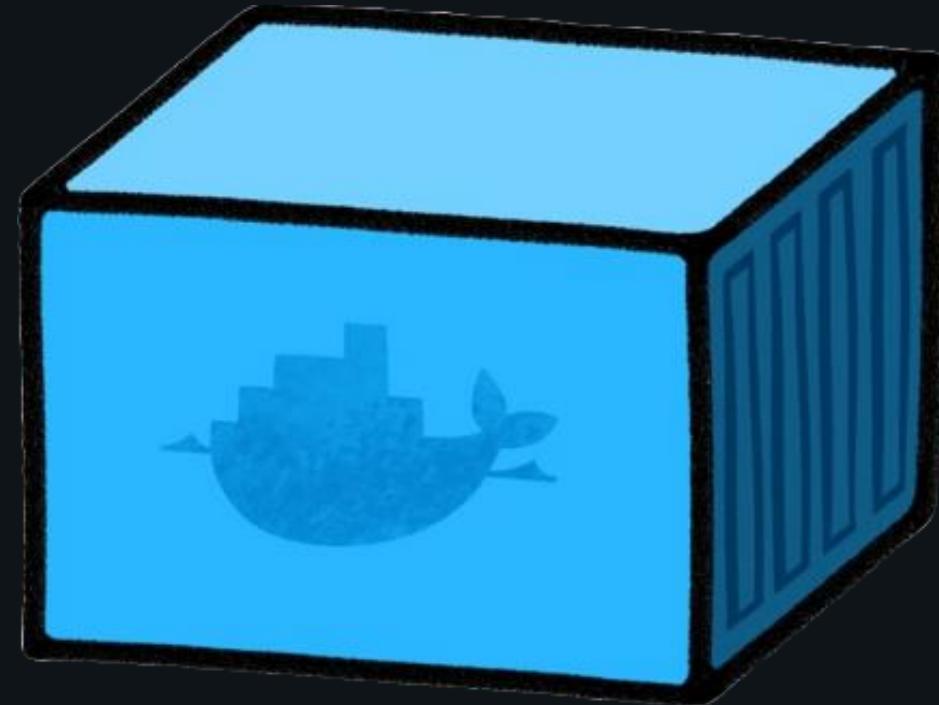
OpenShift



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

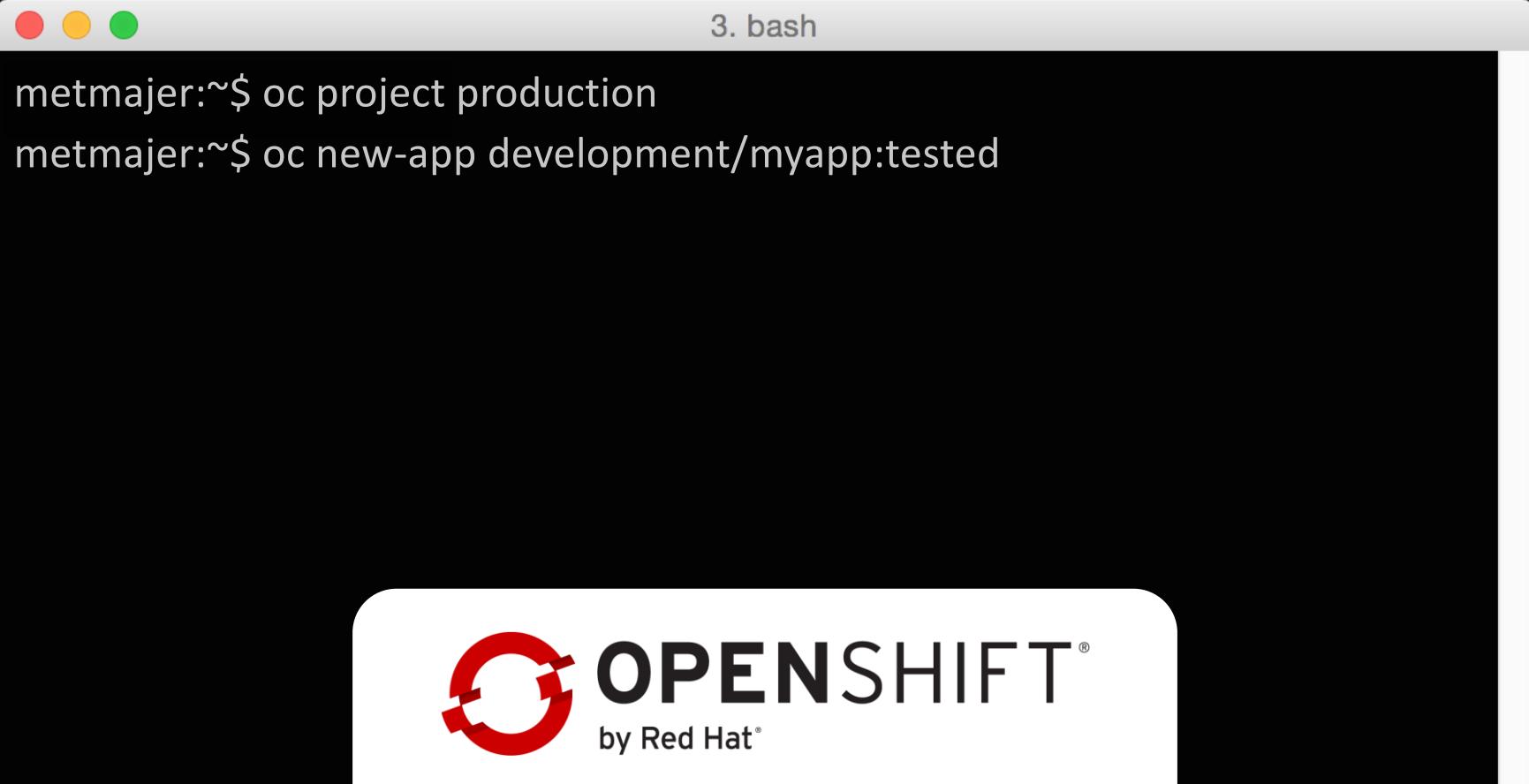
Container API

Container Host

You provide {

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

How to promote an app into production today...



The image shows a screenshot of a terminal window with a dark background. The window title is "3. bash". Inside the terminal, the following commands are displayed:

```
metmajer:~$ oc project production
metmajer:~$ oc new-app development/myapp:tested
```

Below the terminal window, there is a white rectangular badge with rounded corners. It features the OpenShift logo (a red circle with a white arrow) on the left, followed by the word "OPENSHIFT" in a large, bold, black sans-serif font. Below "OPENSHIFT", it says "by Red Hat®" in a smaller, black font.

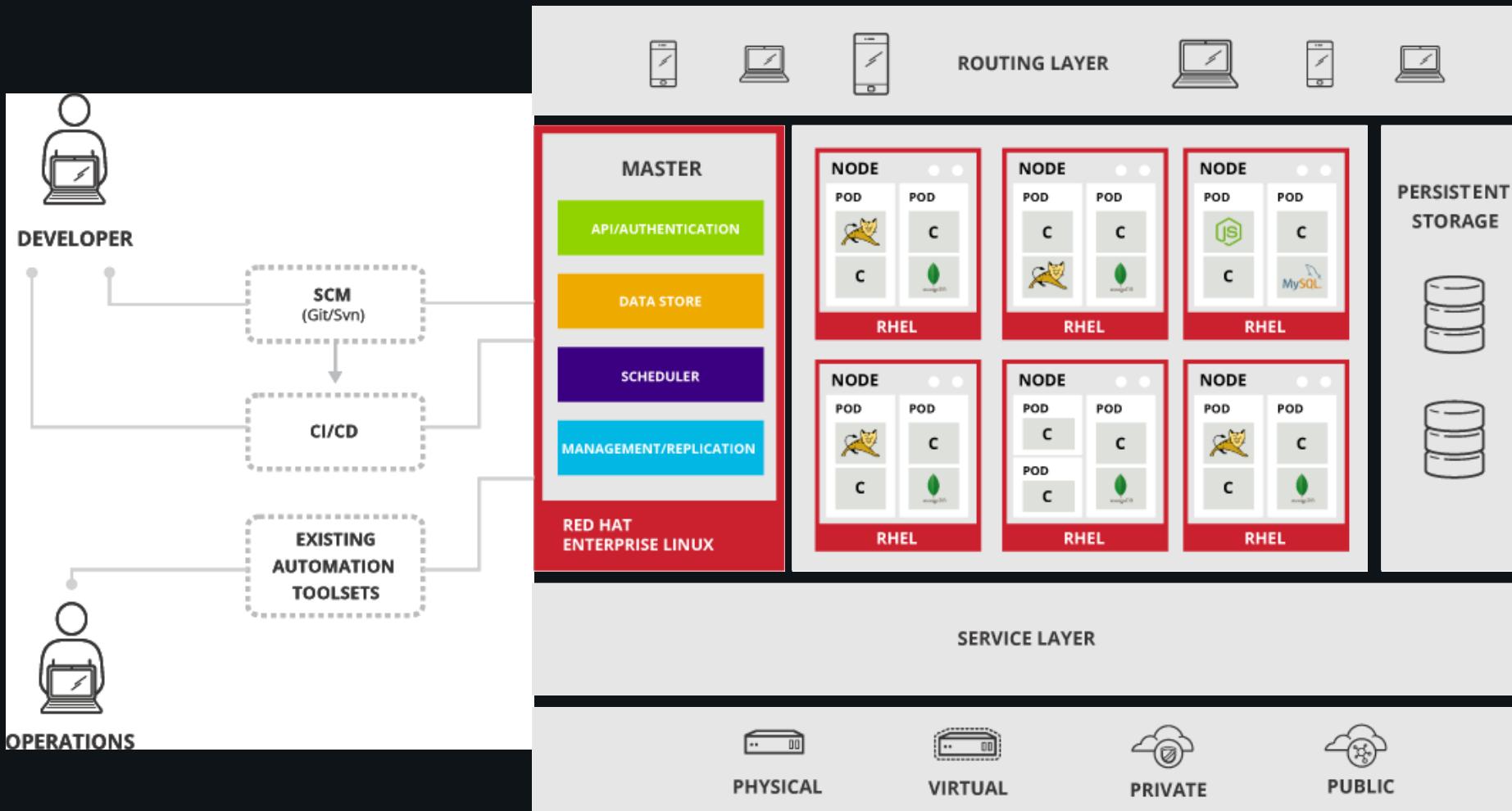
OpenShift = Docker + Kubernetes + UX

Community Powered Innovation



OpenShift Architecture

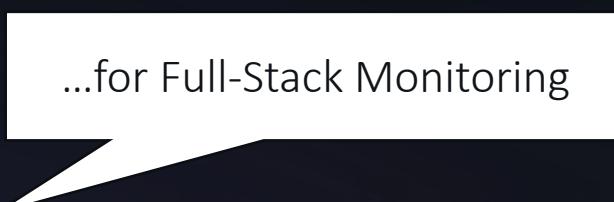
OpenShift Architecture



How do I...



...deploy OneAgent?



...for Full-Stack Monitoring

Option A: Full Stack via OneAgent Operator

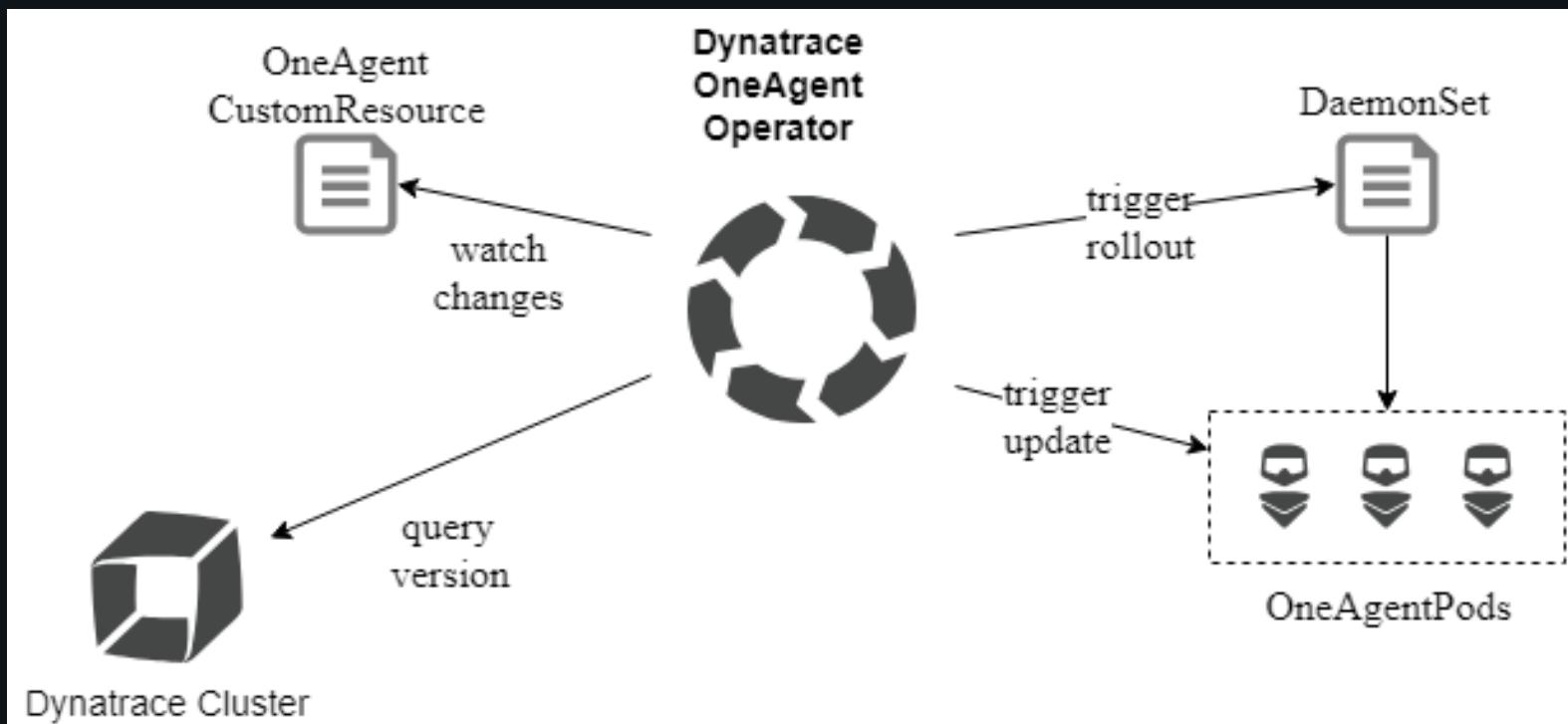
- What is an operator in Kubernetes?
 - Kubernetes version 1.7 introduced the concept of [custom resources and controllers](#), which allow for extending the Kubernetes API. These extension capabilities enable the Kubernetes community to implement domain-specific applications as first-class Kubernetes 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.



Deploy 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 explained 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 <code>ENVIRONMENTID</code> with your Dynatrace environment ID in <code>https://ENVIRONMENTID.live.dynatrace.com/api</code> . Dynatrace Managed: Provide your Dynatrace Server URL (<code>https://<YourDynatraceServerURL>/e/<ENVIRONMENTID>/api</code>)	
tokens	Name of the secret that holds the API and PaaS tokens from above.	Name of custom resource (<code>.metadata.name</code>) 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 <code>.spec.image</code> to <code>registry.connect.redhat.com/dynatrace/oneagent</code> in the custom resource and provide image pull secrets as shown in the next step.	<code>docker.io/dynatrace/oneagent:latest</code> 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 <code>INSTALL_PATH</code> . We recommend to set <code>APP_LOG_CONTENT_ACCESS=1</code>	<code>[]</code>

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`

Deploy 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<br><code>https://ENVIRONMENTID.live.dynatrace.com/api</code> .<br>Dynatrace Managed: Provide your Dynatrace Server URL<br>( <code>https://&lt;YourDynatraceServerURL&gt;/e/&lt;ENVIRONMENTID&gt;/api</code> ) |                                                                     |
| tokens    | Name of the secret that holds the API and PaaS tokens from above.                                                                                                                                                                                                                         | Name of custom resource<br>( <code>.metadata.name</code> ) if unset |
| args      | Parameters to be passed to the OneAgent installer. All the <a href="#">command line parameters of the installer</a> are supported, with the exception of <code>INSTALL_PATH</code> . We recommend to set<br><code>APP_LOG_CONTENT_ACCESS=1</code>                                         | []                                                                  |

## 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?

---



...for PaaS Monitoring

## 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

The screenshot shows the Dynatrace Settings interface. The top navigation bar is dark grey with white text, showing the path: Settings > Cloud and virtualization > AWS & VMware. The main content area has a light grey background. On the left is a sidebar with a dark grey header labeled "Settings". Below it are several sections with dropdown arrows:

- Monitoring
- RUM/Mobile monitoring
- Cloud and virtualization (this section is expanded, showing "Connect vCenter or Amazon account")
- AWS & VMware (this section is expanded, showing "Service detection" and "Log analytics")
- Service detection
- Log analytics

The main content area has a title "Cloud & virtualization" with a cloud icon. It contains a list of instructions:

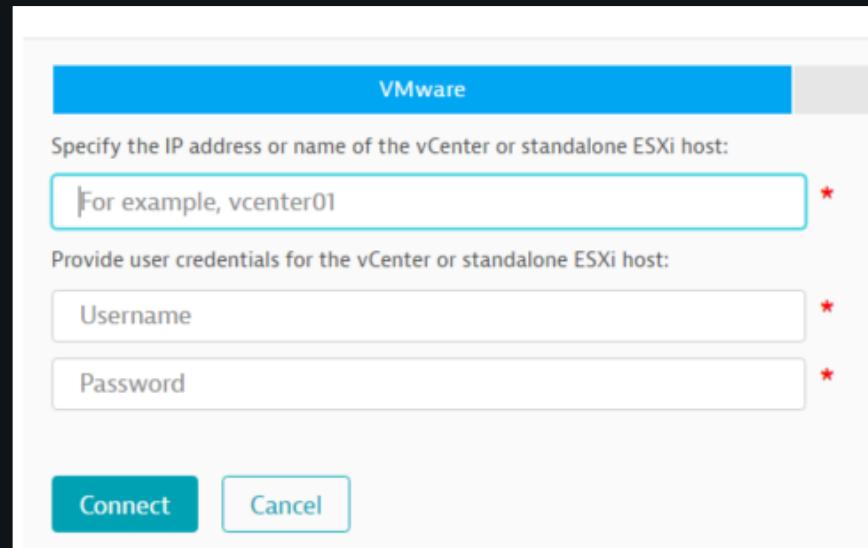
- For VMware instances, connect all vCenter servers that manage virtual machines where Dynatrace runs.
- For AWS instances, go to your Amazon Console and prepare access for Dynatrace using either key or IAM role.

A red note at the bottom states: "Amazon may charge \$0.01 per 1,000 requests for CloudWatch API access - only when the number of requests exceeds the limit of the account you use with Dynatrace." At the bottom of the main content area is a blue button with a plus sign and the text "Connect new instance".

## 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?

---

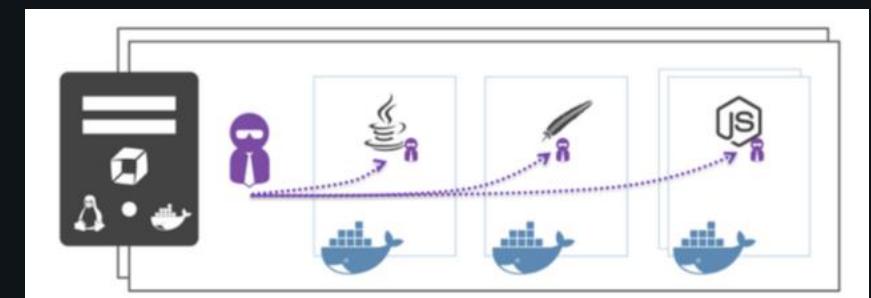
- Dynatrace offers full-featured Docker monitoring, giving you all the same monitoring functionality for Dockerized applications that is available for non-containerized applications.



## 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.

The screenshot shows the 'Monitored technologies' settings page. At the top, there's a header with a gear icon and the title 'Monitored technologies'. Below the header, a sub-instruction reads: 'Specify which technologies Dynatrace should automatically monitor on your hosts. If you only want to enable technology-specific monitoring for individual hosts, disable global monitoring settings on this page and enable them for individual hosts using [host settings](#)'.

The main interface has two tabs at the top: 'Supported technologies' (which is selected) and 'Custom plugins'. The 'Supported technologies' tab displays a list of technologies with their monitoring status and edit options:

| Technology                                                                            | Type             | Monitoring: Off/On               | Edit                 |
|---------------------------------------------------------------------------------------|------------------|----------------------------------|----------------------|
| Plugin manager                                                                        | Dynatrace plugin | <input checked="" type="radio"/> | <a href="#">Edit</a> |
| Configure Couchbase                                                                   | Dynatrace plugin | <input checked="" type="radio"/> | <a href="#">Edit</a> |
| Configure CouchDB                                                                     | Dynatrace plugin | <input checked="" type="radio"/> | <a href="#">Edit</a> |
| Docker containers<br><small>Requires Dynatrace OneAgent version 1.57 or later</small> | Dynatrace plugin | <input checked="" type="radio"/> | <a href="#">Edit</a> |
| Configure Elasticsearch                                                               | Dynatrace plugin | <input checked="" type="radio"/> | <a href="#">Edit</a> |

# Demo

---

# OpenStack

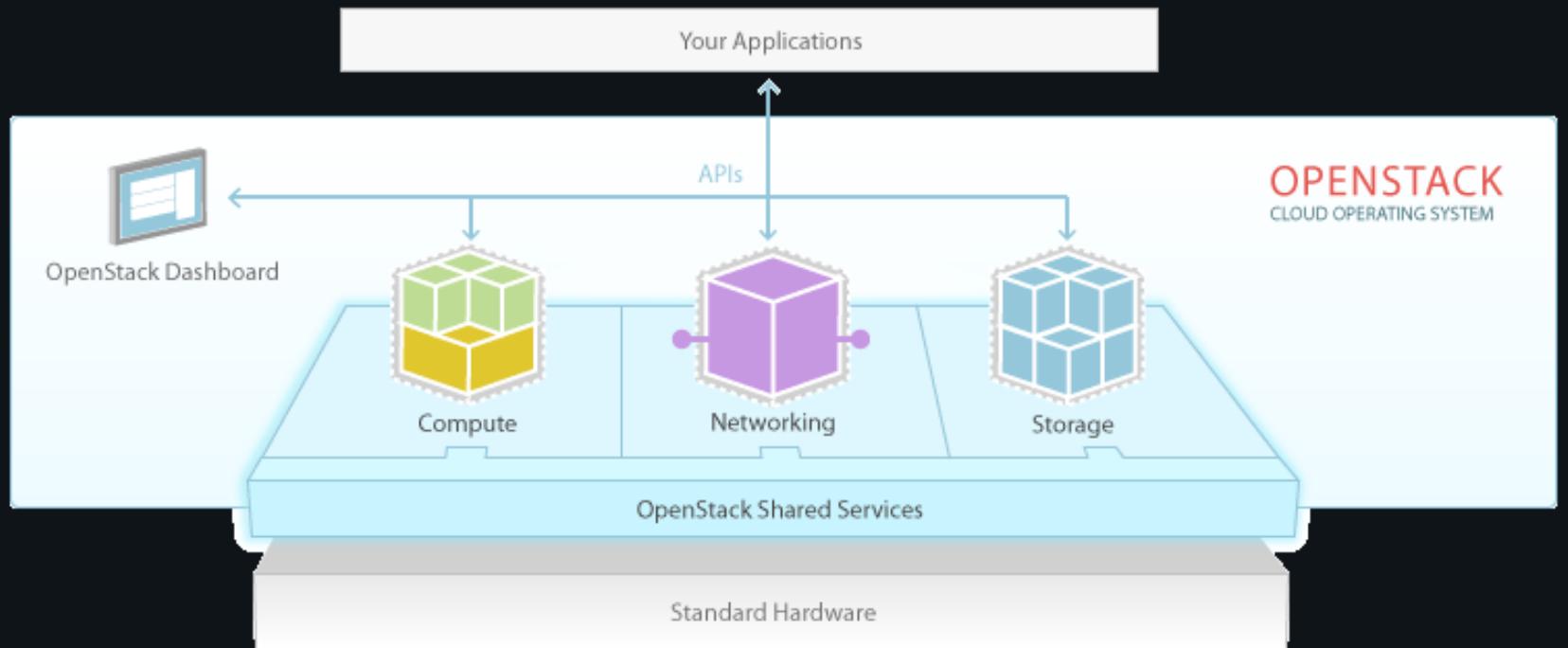
---



# What is OpenStack

---

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



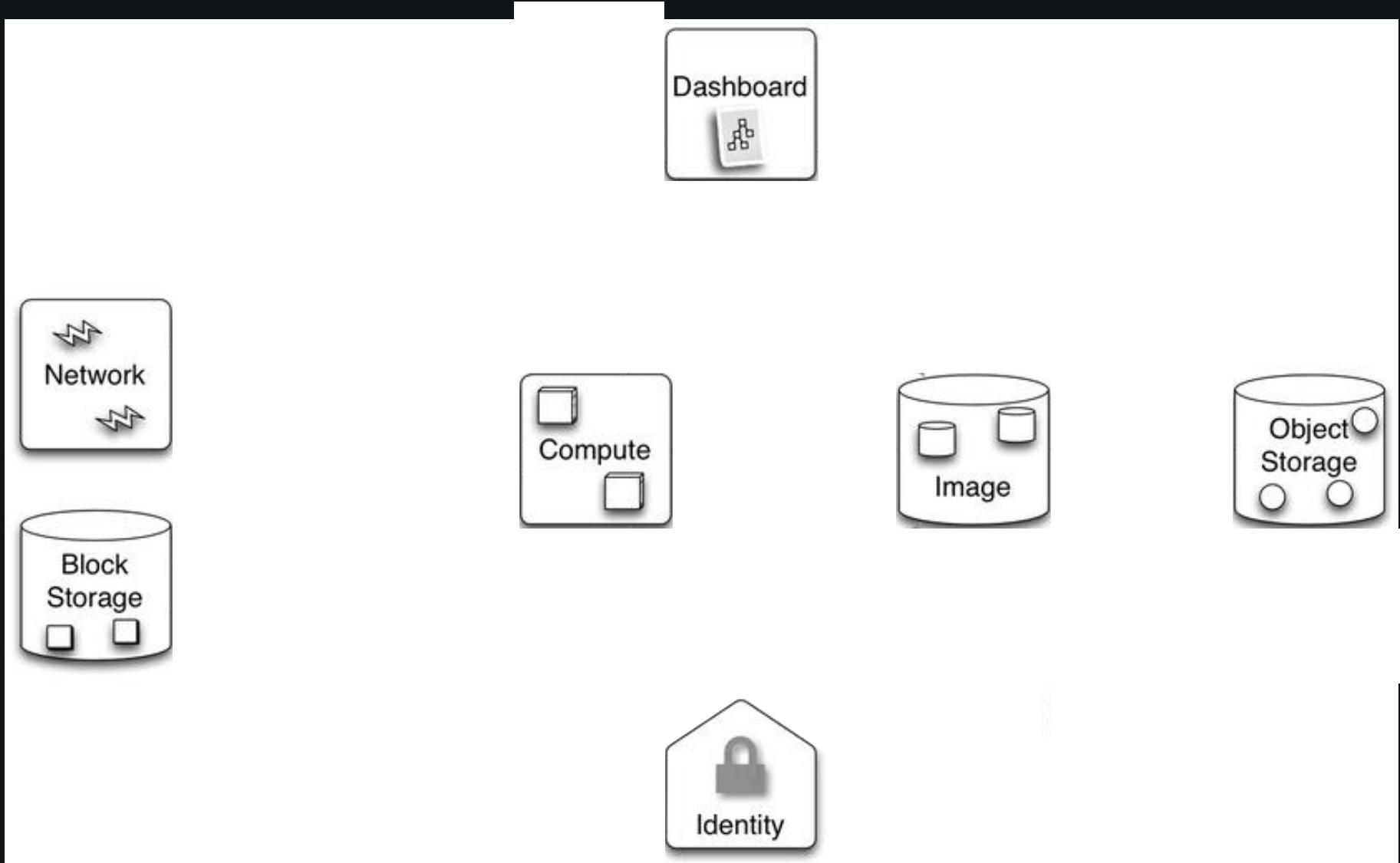
## Details about OpenStack

---

- 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

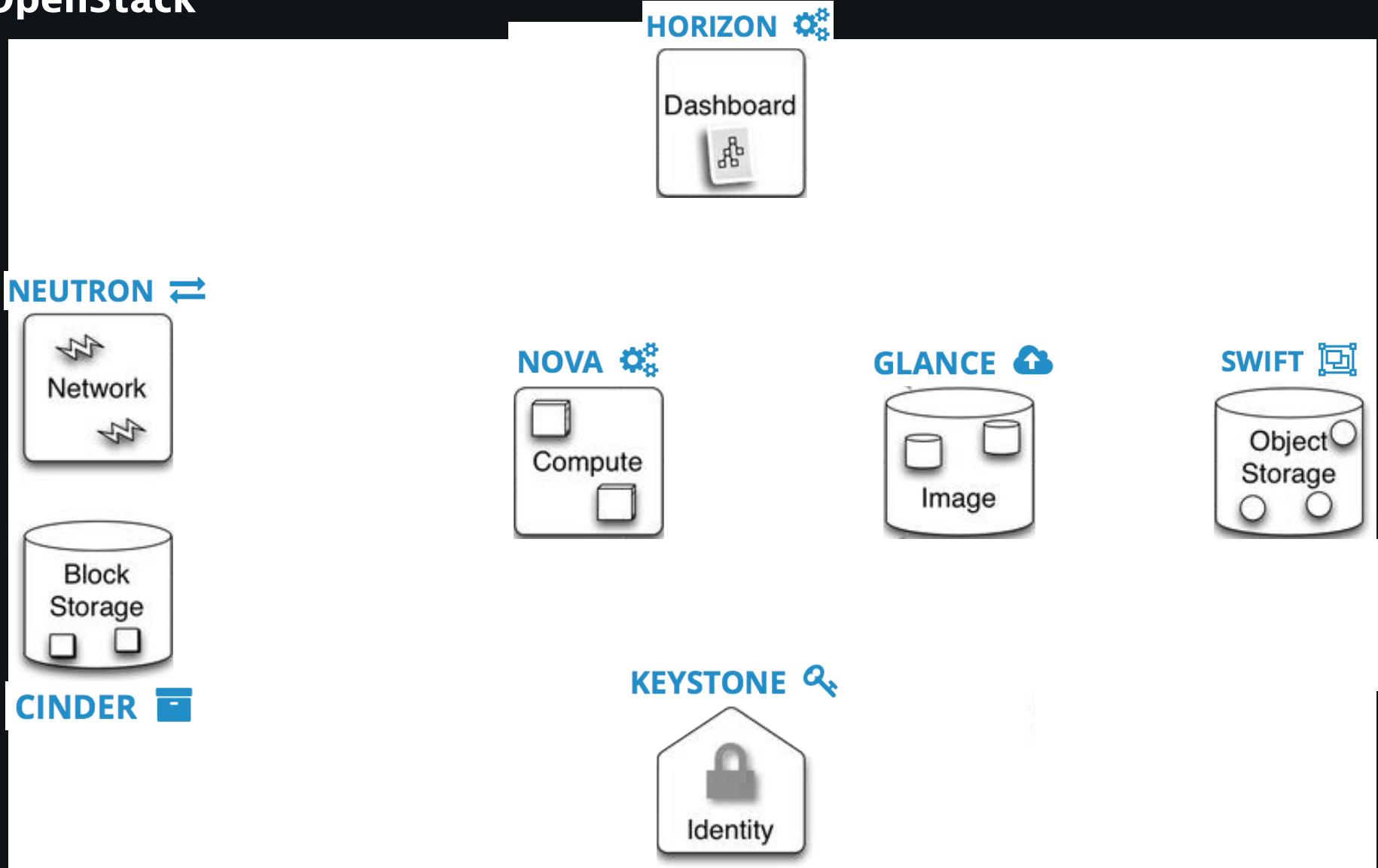
## Details about OpenStack

- Micro service application
- Several core components
- Python



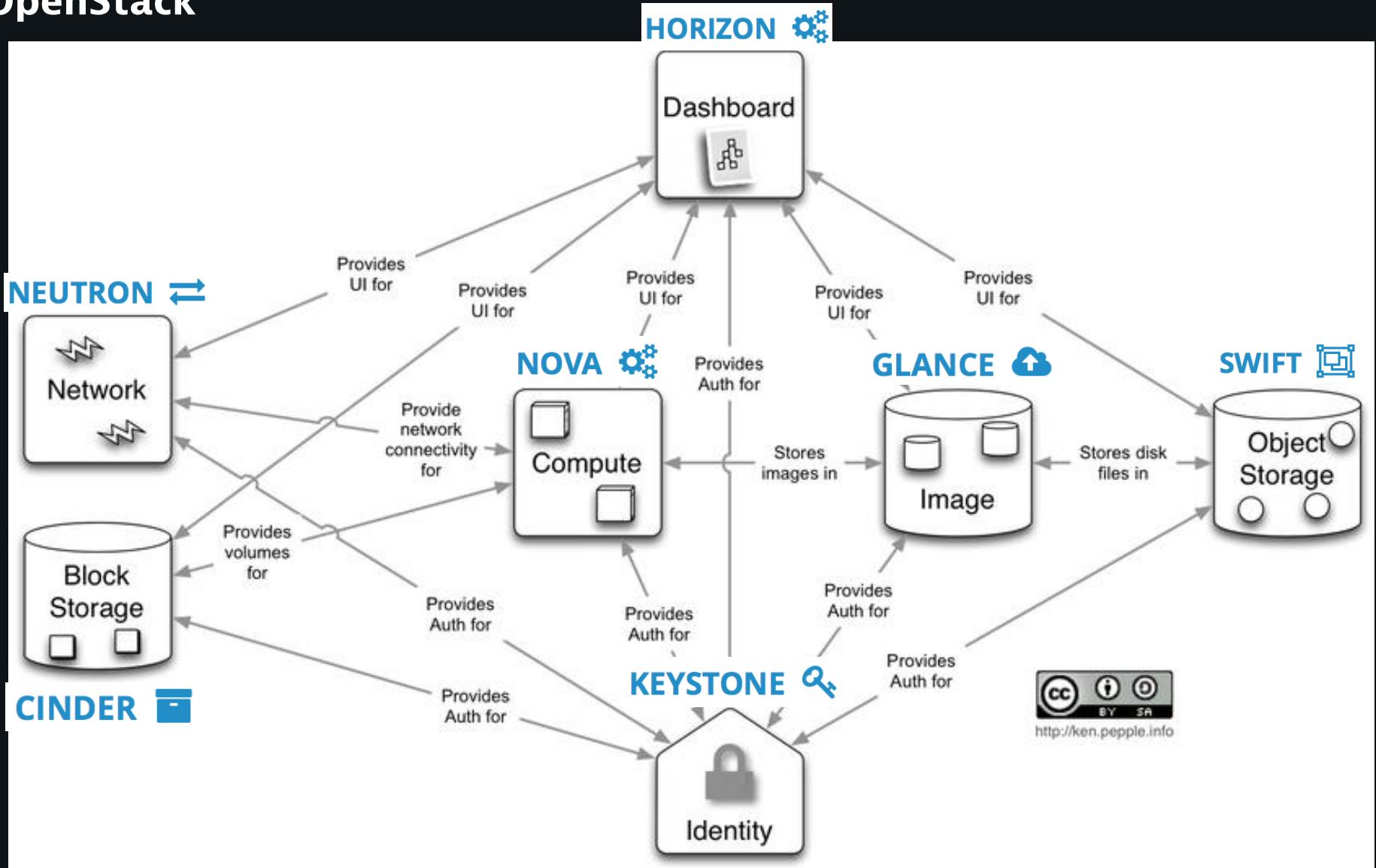
## Details about OpenStack

- Micro service application
- Several core components
- Python



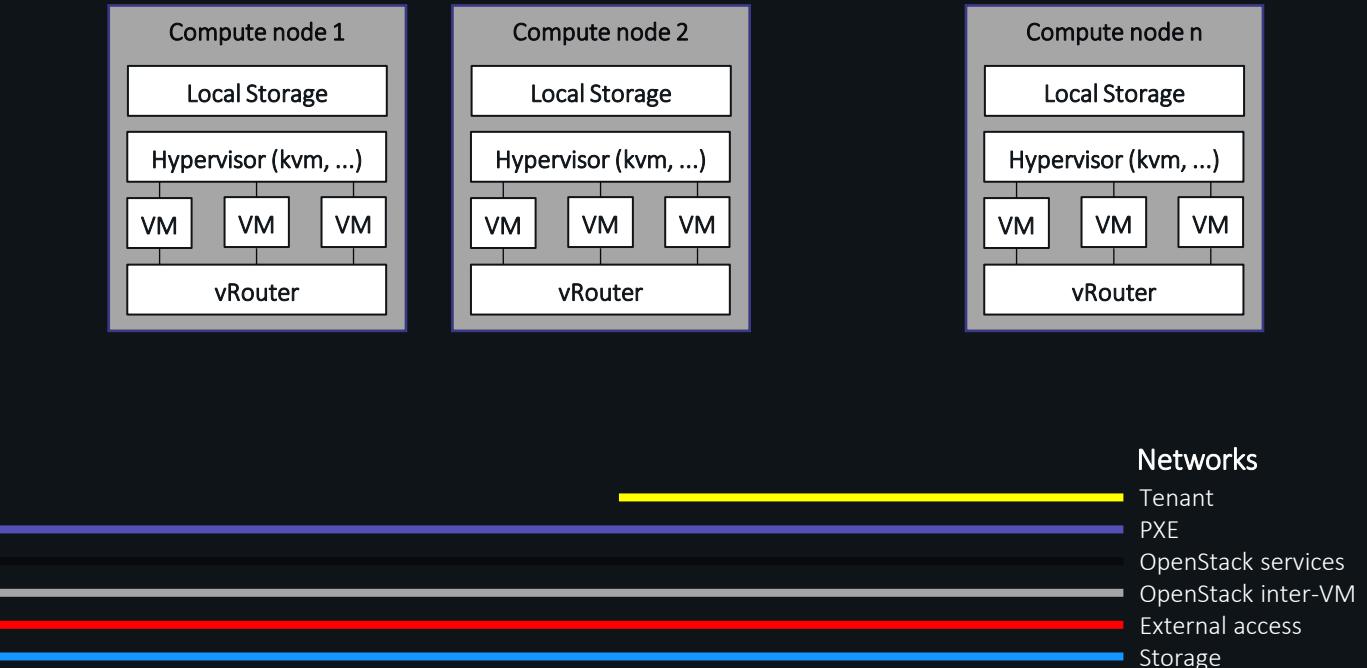
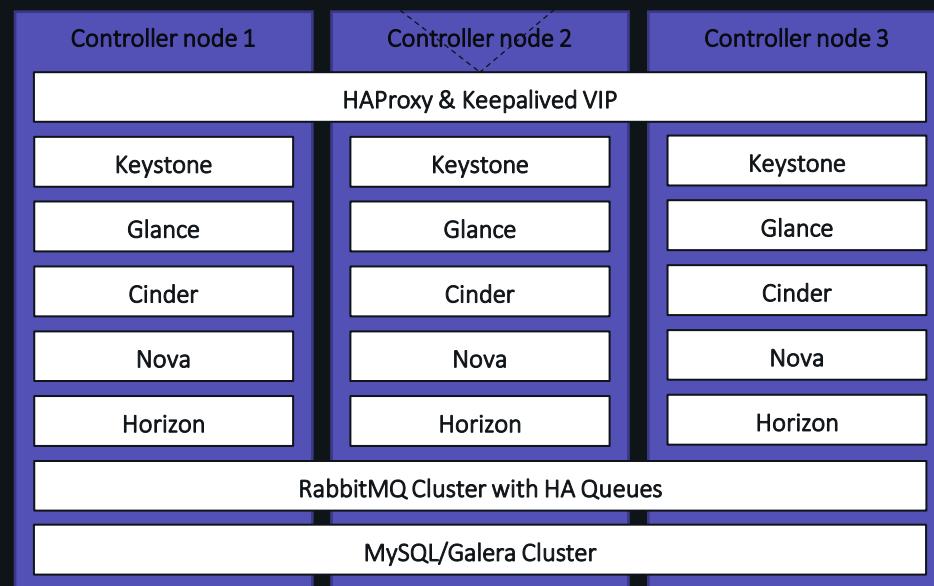
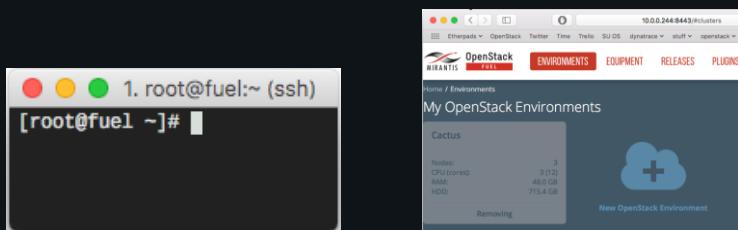
## Details about OpenStack

- Micro service application
- Several core components
- Python



# Details about OpenStack

- Typical OpenStack cluster



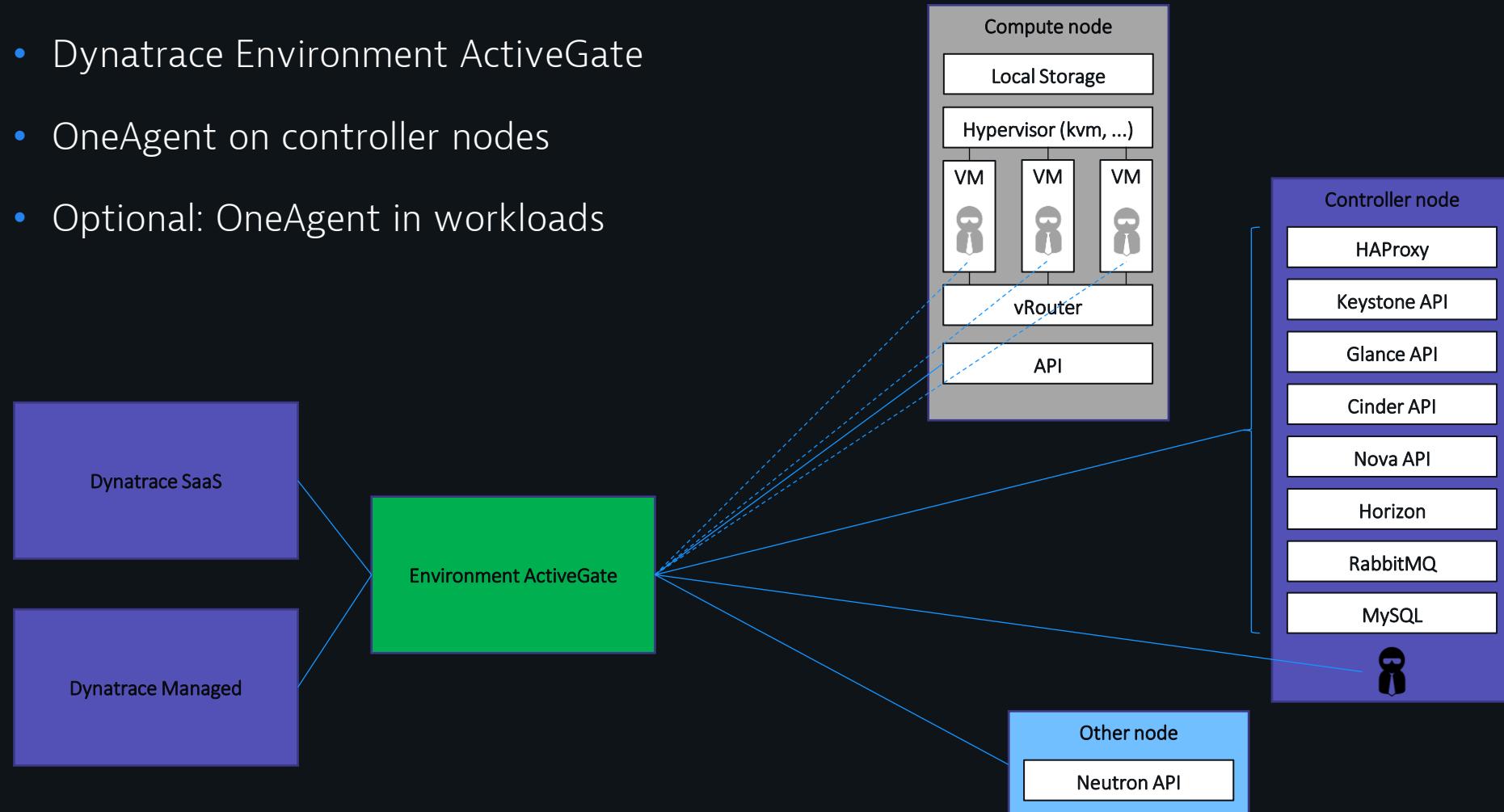
Open vSwitch,  
ML2, Contrail,  
Midokura, ...

External Storage  
(NAS, SAN, SDS, DFS)

- ceph
- 3<sup>rd</sup> party (e.g. NetApp)
- ...

# 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