

GENESYS

This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

Docker Deployment Guide

High Availability

Contents

- 1 High Availability
 - 1.1 High Availability with Amazon Web Service (AWS)
 - 1.2 High Availability with Kubernetes
 - 1.3 Container Orchestration with Kubernetes

High Availability

High Availability with Amazon Web Service (AWS)

Each Genesys product component is ensured high availability by implementing active-active configuration. If you host the product on public cloud (such as AWS), the container cluster is deployed within Auto Scaling Group (ASG). This ensures that a minimum number of specified containers run to serve the traffic.



ASG is also deployed across multiple Availability Zones (AZ) to ensure availability in case of zone failure.



High Availability with Kubernetes

You can deploy a Genesys product in a private cloud or on premise. During such deployment scenarios, containers are implemented on container orchestration platforms such as Kubernetes.



Kubernetes clusters enable a higher level of abstraction to deploy and manage a group of containers that comprise microservices in a cloud-native application. A Kubernetes cluster provides a single Kubernetes API entry point, cluster-wide resource naming scheme, placement engine and scheduler for pods, service network routing domain, and authentication and authorization model.

Kubernetes can be deployed in the following scenarios:

- Across multiple availability zones, but within a single cloud provider Cloud provider services like storage, load balancers, network routing may interoperate easily.
- In the same geographical region Network performance will be fast enough to act like a single data center.
- Across multiple geographical regions High network cost and poor network performance may be prohibitive.
- Across multiple cloud providers with incompatible services and limited interoperability Inter-cluster

networking will be complex and expensive to set up in an efficient way.

Container Orchestration with Kubernetes

Auto-scaling with Kubernetes

Kubernetes pods can be auto-scaled up or down based on metrics such as CPU and memory utilization. Kubernetes implements Horizontal Pod Autoscaler (HPA) to achieve auto-scaling. HPA collects metrics about pods from Heapster. Based on the auto-scaling rule, HPA either increases or decreases the number of pods through the deployment object created for the Genesys microservice. The following diagram illustrates the auto-scaling architecture with HPA and Heapster.



Kubernetes Setup with Rancher

Rancher provides an easier process to set up a Kubernetes cluster either on cloud or on premise. It hides the complex steps involved in setting up the master and worker nodes. Rancher is recommended for a push button setup of Kubernetes cluster for development and QA purposes. However, it is not recommended for production use. If you have any queries regarding Rancher, contact the Genesys Engage Architecture team

(pureengagearchitecture@genesyslab.onmicrosoft.com).

To set up a single node Kubernetes cluster with Rancher, follow these steps:

- Start Rancher Server by executing the following command. sudo docker run -d --restart=unless-stopped -p 8080:8080 rancher/server
- 2. Open the Rancher console using **<ip>:8080** on the browser.

3. On the **Default** menu, select **Manage Environments**.

177	Environment	STACKS	✓ CATALOG ✓	INFRASTRUCTURE 🗸	ADMIN 🗸 🕄		
\wedge	All Environments		rovice or launching a	container you'll need to a	d a Linux host v	/ith a supported version of Docker. Add a host	
	🛆 Default	ĩ	i vice of faultering a	container, you inneed to a	JU B EIHUX HOSE I	ner aupportee version of Docker Add a nose	
Add	Edit "Default"						
Addi	Manage Environm	ients					

4. Click Add Environment.

O De	fault V STACKS V	CATALOG V	INFRASTRUCTURE V		API 🗸			
Before	adding your first servi	ce or launching a	container, you'll need to ad	dd a Linux host wit	h a supported version of Dock	er. Add a host		
ironmei	Add Environmen							
ancher supp	orts grouping resourc	es into multiple <mark>e</mark>	nvironments. Each one get	ts its own set of ser	vices and infrastructure reso	irces, and is owned by one or more users or	groups.	
ancher supp or example,	orts grouping resourc you might create sepa	es into multiple <mark>e</mark> rate "dev", "test", a	nvironments. Each one get and "production" environm	ts its own set of ser nents to keep thing	vices and infrastructure reso s isolated from each other, an	rrces, and is owned by one or more users or d give "dev" access to your entire organizati	groups. on but restrict the "produ	uction"
ancher supp or example, ovironment	orts grouping resourc you might create sepa to a smaller team.	es into multiple e r rate "dev", "test", a	nvironments. Each one get and "production" environm	ts its own set of ser nents to keep thing	vices and infrastructure reso s isolated from each other, an	urces, and is owned by one or more users or d give "dev" access to your entire organizati	groups. Ion but restrict the "produ	uction"
ancher supp or example, avironment	iorts grouping resourc you might create sepa to a smaller team.	es into multiple er rate "dev", "test", a	nvironments. Each one get and "production" environm	ts its own set of ser nents to keep thing	vices and infrastructure reso s isolated from each other, an	urces, and is owned by one or more users or d give "dev" access to your entire organizati	groups. on but restrict the "produ	uction"
ancher supp or example, nvironment	oorts grouping resourc you might create sepa to a smaller team.	es into multiple e r rate "dev", "test", a	nvironments. Each one get and "production" environm	ts its own set of ser nents to keep thing	vices and infrastructure reso	irces, and is owned by one or more users or d give "dev" access to your entire organizati	groups. on but restrict the "produ	uction"
ancher supp or example, avironment	orts grouping resourc you might create sepa to a smaller team. Name \$	es into multiple er rate "dev", "test", a	nvironments. Each one get and "production" environm Description \$	ts its own set of ser nents to keep thing	vices and infrastructure reso s isolated from each other, an Template \$	irces, and is owned by one or more users or d give "dev" access to your entire organizati Orchestration \$	groups. on but restrict the "produ Default	uction"
ancher supp or example, avironment State \$	oorts grouping resourc you might create sepa to a smaller team. Name \$ Default	es into multiple er	nvironments. Each one get and "production" environm Description \$ No description	ts its own set of ser	vices and infrastructure reso s isolated from each other, an Template Cattle	rrces, and is owned by one or more users or d give "dev" access to your entire organizati Orchestration ≎ Cattle	groups. on but restrict the "produ Default	uction"

5. On the screen displayed, create a new environment by providing a name in the **Name** field. Then, select **Kubernetes** as the orchestration for the new environment.

1						API↓
\land	Before adding y	our first servic	e or launching a c	ontainer, you'll need to ad	ld a Linux host v	with a supported version of Docker. Add a host
Add	Environment					
N	lame					Description
	test					e.g. Environment for developer experimentation
E	nvironment Templ	ate				
				Cattle	Kuberne	
						Orchestration: Kubernetes
					Framewo	rk: Network Services, Healthcheck Service
						Networking: Rancher IPsec

This starts setting up Kubernetes.



- 6. Complete the wizard.
- 7. On the **INFRASTRUCTURE** menu, select **Hosts**, and then click **Add Host**.

<table-cell> Rancher</table-cell>	× 🗅 API: sdnc-ha	×		ExII	-		×
$\epsilon \rightarrow c$) 10.147.132.10:8880/env/1a8/infr	a/hosts			\$	• •) : C
🔛 Apps 📙 co	onfluence 🛄 info 🛄 tools 🛄 a	tt 🛄 onap 🛄 ME					
FT 0	sdna ha 🗸 🔰 KUBERNETES 🗸	INFRASTRUCTURE ~	admin 🗸 🥵 🌐 Api 🗸				
Bef	ore adding your first service or	Hosts	u'll need to add a Linux host with a supported version of Docker. Add a host				
Hosts A	dd Host	Storage		Show Sy	rstem	•••	R
		Secrets Certificates	No hosts or containers yet.				
		Registries					

8. Enter the IP of the same host (for single node cluster) or a different host (for multi-node cluster). Hosts: Add Host

	Custom Image available machine drivers
1 2 3	Start up a Linux machine somewhere and install a supported version of Docker on it. Make sure any security groups or firewalls allow traffic: • From and To all other hosts on UOP ports 500 and 4500 (for IPsec networking) Optional: Add labels to be applied to the host. Add Label
4	Specify the public IP that should be registered for this host. If left empty, Rancher will auto-detect the IP to use. This generally works for machines with unique public IPs, but will not work if the machine is behind a firewall/NAT or if it is the same machine that is running the rancher/server container.
5	Copy, paste, and run the command below to register the host with Rancher: sudo docker run -e CATTLE_AGENT_IP-"1.1.1.1"rm -privIleged -v /var/run/docker.sock:/var/fun/docker.sock -v /var/fib/rancher_/var/fib/ rancher rancher/agent:v1.2.10 http://34.244.174.231:BBBN/v1/scripts/CB7162BE0FE126EED918:15140784090001;BdPBbus 33486ABDeFGscrauf

- 9. Execute the command generated by the wizard on the node to be added.
- 10. Watch the status of the Kubernetes cluster getting initialized and the services getting started.
- 11. Navigate to the cluster environment and select CLI. The command line interface that interacts with the

cluster using Kubectl command will be accessible from the browser.

12. To interact with the cluster from a different computer, copy the config file that Rancher generates and paste it on the ~/.kube/config file. Create the config file if it is not available.

Kubernetes Helm for Package Management

Helm is a package manager for Kubernetes. Helm packages multiple K8s resources into a single logical deployment unit called Chart. Helm is also a deployment manager for Kubernetes. It helps in:

- Performing repeatable deployments
- Managing (reusing and sharing) dependencies
- Managing multiple configurations
- Updating, rolling back, and testing application deployments (releases)

Here are the components of a typical Helm environment:

- Chart a package; bundle of Kubernetes resources
- Release a chart instance that is loaded into Kubernetes
- Repository a repository of published charts
- Template a K8s configuration file with Go template

[root@ip-172-30-0-249 testapi-chart]# tree
<pre></pre>
NOTES.txt service.yaml values.yaml

The Genesys Kubernetes manifests are packaged as Helm charts. The charts are in .tgz format. You can download the charts from the Helm repository that will be created for each project, and then install them. The following example illustrates the steps to be followed while installing a chart from the repository.

- 1. Download the chart from the **git** repository using the wget command.
- 2. Install the chart using the helm install command as shown in the following image.

[root@i	gws helm]# ls
ws-core-auth-chart-0.	1,0.tgz
ws-core-environment-c	chart-0.1.0.tgz
ws-elasticsearch-char	rt-0.1.0.tgz
	ation-chart-0.1.0.tgz
ws-platform-ocs-chart	t-0.1.0.tgz
ws-platform-statistic	cs-chart-0.1.0.tgz
jws-platform-voice-cha	art-0.1.0.tgz
gws-postgres-chart-0.1	1.0.tgz
jws-redis-cluster-char	rt-0.1.0.tgz
testapi	
[root@ 170 00 0 010	gws_helm]# helm install gws-core-auth-chart-0.1.0.tgz

For detailed information about Helm installation, refer to https://github.com/kubernetes/helm/.