

Installing Automic Automation Kubernetes Edition v21

How to deploy to Google Cloud Platform (GCP)

Version 1.1

Broadcom, the pulse logo, and Connecting everything are among the trademarks of Broadcom and/or its affiliates in the United States, certain other countries, and/or the EU.

Copyright © 2021 by Broadcom. All Rights Reserved.

The term “Broadcom” refers to Broadcom Inc. and/or its subsidiaries. For more information, please visit www.broadcom.com.

Broadcom reserves the right to make changes without further notice to any products or data herein to improve reliability, function, or design. Information furnished by Broadcom is believed to be accurate and reliable. However, Broadcom does not assume any liability arising out of the application or use of this information, nor the application or use of any product or circuit described herein, neither does it convey any license under its patent rights nor the rights of others.

Contents

Chapter 1: Introduction	4
Chapter 2: Create the PostgreSQL Database for Automic	5
Chapter 3: Create the Secrets.....	7
3.1 Automic ImagePullSecret used to retrieve the images from GCR.....	7
3.2 DB secret with the connection information.....	7
3.3 Client 0 secret with pre-set credentials	7
Chapter 4: Customize the JCP-WS and AWI Backend services	8
Chapter 5: Deploy AAKE in the GKE Cluster.....	9
5.1 Download the AAKE zip package and install the Automic Helm Plugin and Helm chart	9
5.2 Customize values.yaml for GKE.....	9
5.3 Configure the gcloud CLI to connect to the cluster via command line.....	9
5.4 Install AAKE using Helm	10
Chapter 6: Configure TLS certificates.....	11
Chapter 7: Expose the Cluster to the outside world	12
Chapter 8: Connect Agents via HTTPS Load Balancer.....	14
8.1 Connect TLS-enabled agents.....	14
8.2 Connect non-TLS agents via TLS Gateway	14

Chapter 1: Introduction

The Google Kubernetes Engine (GKE) offered by Google Cloud Platform (GCP) can be used to deploy and manage the Automic Automation Kubernetes Edition.

If you are new to GCP and Kubernetes, the [GKE quickstart guides](#) can provide you with all the information you need, starting with how to use the Google Cloud Shell to work with the cluster, to making the most of Kubernetes with autoscaling. You can also use the [Google Cloud SDK](#) to connect to your cluster and deploy AAKE

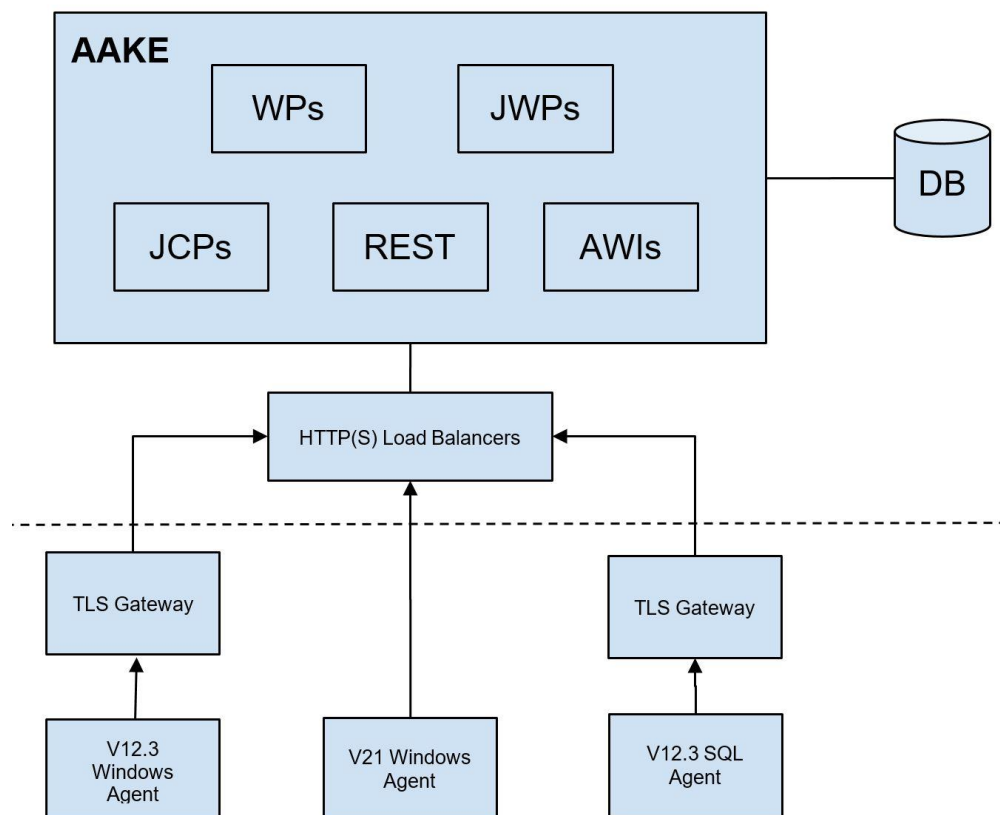
To connect from the GKE cluster to the PostgreSQL database instance you can either use a private IP address or a public one by configuring the [Cloud SQL Auth proxy](#). Since the second option is not supported with AAKE, a private IP address is used to connect Automic processes to the managed database.

GCP prerequisites at a glance:

- Kubernetes cluster on GKE
- Static public IP address
- Cluster node pool with autoscaling enabled
- PostgreSQL database instance

Be aware, this document does not replace the Automic documentation or a basic understanding of Kubernetes concepts and other Cloud relevant components, such as Load Balancers.

The following is only an example of how to deploy AAKE on GKE. and connect both TLS and non-TLS agents. For this purpose, TLS Gateways in CP mode are used to connect version 12.3 agents, while new TLS agents are able to communicate with the JCPs via HTTPS Load Balancers.. An overview can be found in the diagram below.



Chapter 2: Create the PostgreSQL Database for Automic

The PostgreSQL instance must have the `vacuum_cost_limit` flag set to 10000 to load the Automic database. This can be set while creating the instance or updated afterwards. The other mandatory and recommended settings are provided in the Automic documentation.


You can improve the performance of the database connection and Kubernetes by creating both in the same zone on GCP.

You can connect via `gcloud` from the Google Cloud Shell to create the user and database required for AAKE:









```
$ gcloud sql connect oab-db --user=postgres --quiet
```

Alternatively, if the database instance can be accessed from an external network, `psql` or `pg4Admin` can be used.

This requires that the instance is reachable via a public IP and also that the IP addresses of the requesting hosts are included in the authorized networks.

 SQL

PRIMARY INSTANCE

-  Overview
-  Query Insights
-  Connections
-  Users
-  Databases
-  Backups
-  Replicas
-  Operations

Connections

All instances > oab-db

✔ **oab-db**

PostgreSQL 12

[NETWORKING](#) SECURITY

Choose how you want your source to connect to this instance, then define which networks are authorized to connect. [Learn more](#)

You can use the Cloud SQL Proxy for extra security with either option. [Learn more](#)

Instance IP assignment

Private IP
Assigns an internal, Google-hosted VPC IP address. Requires additional APIs and permissions. Can't be disabled once enabled. [Learn more](#)

Associated networking
Select a network to create a private connection

Network *
default

✔ Private services access connection for network **default** has been successfully created. You will now be able to use the same network across all your project's managed services. If you would like to change this connection, please visit the [Networking page](#).

[SHOW ALLOCATED IP RANGE OPTION](#)

Public IP
Assigns an external, internet-accessible IP address. Requires using an authorized network or the Cloud SQL Proxy to connect to this instance. [Learn more](#)

Authorized networks
You can specify CIDR ranges to allow IP addresses in those ranges to access your instance. [Learn more](#)

Your Hostname (198.51.100.1)	▼
Your Hostname (203.0.113.1)	▼
ADD NETWORK	

App Engine authorization
All apps in this project are authorized by default. You can use [Cloud IAM](#) to authorize apps in other projects. [Learn more](#)

[SAVE](#) [DISCARD CHANGES](#)

```
CREATE USER "oab" WITH LOGIN PASSWORD 'dev' CONNECTION LIMIT -1;

CREATE DATABASE ae WITH OWNER = "oab" TEMPLATE = template0 ENCODING = 'UTF8'
LC_COLLATE = 'C' LC_CTYPE = 'C' CONNECTION LIMIT = -1;

CREATE SCHEMA dbo AUTHORIZATION "oab";
ALTER ROLE "oab" IN DATABASE ae SET search_path TO 'dbo';
```

Chapter 3: Create the Secrets

Sensitive information relevant to the Automic system is stored in secrets and retrieved during deployment. You must download a json file that stores the credentials required to pull the container images from the [Automic Downloads Page](#).

3.1 Automic ImagePullSecret used to retrieve the images from GCR

```
$ kubectl create secret docker-registry automic-image-pull-secret \  
--docker-server=gcr.io \  
--docker-username=_json_key \  
--docker-password="$(cat ./automic-image-pull-secret.json)" \  
--docker-email=broadcom-com@esd-automic-saas.iam.gserviceaccount.com
```

3.2 DB secret with the connection information

```
$ kubectl create secret generic ae-db \  
--from-literal=host=<your-db-instance-ip-address> \  
--from-literal=vendor=postgres --from-literal=port='5432' --from-literal=user=oab \  
--from-literal=db=ae --from-literal=password=dev \  
--from-literal=data-tablespace-name=pg_default \  
--from-literal=index-tablespace-name=pg_default \  
--from-literal=additional-parameters="connect_timeout=10 client_encoding=LATIN9"
```

3.3 Client 0 secret with pre-set credentials

```
$ kubectl create secret generic client0-user \  
--from-literal=client='0' \  
--from-literal=user=ADMIN \  
--from-literal=department=ADMIN \  
--from-literal=password=admin
```

Chapter 4: Customize the JCP-WS and AWI Backend services

Certain back-end services exposed outside the cluster require additional configuration to optimize their functionality. This can be achieved by creating BackendConfig objects that overwrite default timeouts or enable the usage of cookies.

```
$ kubectl apply -f jcp-ws-backendconfig.yaml
```

```
$ kubectl apply -f awi-backendconfig.yaml
```

jcp-ws-backendconfig.yaml:

```
apiVersion: cloud.google.com/v1beta1
kind: BackendConfig
metadata:
  name: jcp-ws-backendconfig
spec:
  timeoutSec: 86400
  connectionDraining:
    drainingTimeoutSec: 3600
```

awi-backendconfig.yaml:

```
apiVersion: cloud.google.com/v1beta1
kind: BackendConfig
metadata:
  name: awi-backendconfig
spec:
  timeoutSec: 86400
  connectionDraining:
    drainingTimeoutSec: 120
  sessionAffinity:
    affinityType: "GENERATED_COOKIE"
    affinityCookieTtlSec: 3600
  healthCheck:
    checkIntervalSec: 30
    requestPath: /awi/health
```


Chapter 5: Deploy AAKE in the GKE Cluster

The AAKE zip package that can be downloaded from <https://downloads.automic.com> contains a Helm Plugin mainly used to check the status of the installation and a Helm chart with the values.yaml file as the entry point for the configuration.

5.1 Download the AAKE zip package and install the Automic Helm Plugin and Helm chart

```
$ tar xvf automic-automation-plugin-1.0.0.tgz
$ helm plugin install automic-automation-plugin

$ tar xvf automic-automation-1.0.0.tgz
$ cp automic-automation/values.yaml values.yaml
```

5.2 Customize values.yaml for GKE

Add annotations for AAKE services to use Ingress and configure health checks.

```
awi:
  service:
    annotations:
      cloud.google.com/neg: '{"ingress": true}'
      beta.cloud.google.com/backend-config: '{"default": "awi-backendconfig"}'

jcp-rest:
  service:
    annotations:
      cloud.google.com/neg: '{"ingress": true}'

jcp-ws:
  service:
    annotations:
      cloud.google.com/neg: '{"ingress": true}'
      cloud.google.com/app-protocols: '{"ws": "HTTPS"}'
      beta.cloud.google.com/backend-config: '{"default": "jcp-ws-backendconfig"}'
```

5.3 Configure the gcloud CLI to connect to the cluster via command line

```
$ gcloud container clusters get-credentials oab-aake --zone europe-central2-c --
project demos-esd-automic
```

5.4 Install AAKE using Helm

```
$ helm install aake automic-automation-1.0.0.tgz -f values.yaml
```

The screenshot shows the Google Cloud Kubernetes Engine console. On the left is a navigation menu with options like Clusters, Workloads, Services & Ingress, Applications, Configuration, Storage, Object Browser, Migrate to containers, and Config Management. The main area is titled 'Workloads' and shows filters for Cluster 'oab-aake' and Namespace 'v21'. Below the filters, there's a table of workloads. The table has columns for Name, Status, Type, Pods, Namespace, and Cluster. All workloads listed have a status of 'OK'.

<input type="checkbox"/>	Name ↑	Status	Type	Pods	Namespace	Cluster
<input type="checkbox"/>	ae-cp	✓ OK	Deployment	0/0	v21	oab-aake
<input type="checkbox"/>	ae-wp	✓ OK	Deployment	4/4	v21	oab-aake
<input type="checkbox"/>	awi	✓ OK	Deployment	2/2	v21	oab-aake
<input type="checkbox"/>	cust-aa-21-0-0-data-1638198231188	✓ OK	Job	0/1	v21	oab-aake
<input type="checkbox"/>	cust-aa-21-0-0-ready-1638198231188	✓ OK	Job	0/1	v21	oab-aake
<input type="checkbox"/>	initial-data-21-0-0-1638198231188	✓ OK	Job	0/1	v21	oab-aake
<input type="checkbox"/>	install-operator	✓ OK	Deployment	1/1	v21	oab-aake
<input type="checkbox"/>	jcp-rest	✓ OK	Deployment	1/1	v21	oab-aake
<input type="checkbox"/>	jcp-ws	✓ OK	Deployment	3/3	v21	oab-aake
<input type="checkbox"/>	jwp	✓ OK	Deployment	2/2	v21	oab-aake

The Automic Helm plugin can be used to check the status of the installation:

```
$ helm automic-automation status
```

Chapter 6: Configure TLS certificates

You must have valid certificates in place to connect TLS-enabled agents to the AAKE cluster. The TLS handshake is performed between the agent and the HTTPS Load Balancer and there is no need to additionally configure the JCP as is the case for on-prem installations.

The TLS-enabled agents use hostname verification, so make sure the domains of the HTTPS Load Balancers are included as SANs in the certificate if you don't use a wildcard domain.

A ManagedCertificate object can be used to create a certificate including all the required domain names. In this example, the already available static IP address is used with .nip.io to allow the usage of separate domains for AWI, JCP-REST and JCP_WS.

```
$ kubectl apply -f managed-certificate.yaml
```

managed-certificate.yaml:

```
apiVersion: networking.gke.io/v1beta2
  kind: ManagedCertificate
  metadata:
    name: aake-cert-default
  spec:
    domains:
      - awi.<your static public ip address>.nip.io
      - rest.<your static public ip address>.nip.io
      - ws.<your static public ip address>.nip.io
```

The generated certificate is signed by the GlobalSign public Certificate Authority and looks as below:

Network services	Certificate details												
<ul style="list-style-type: none"> Load balancing Cloud DNS Cloud CDN Cloud NAT Traffic Director Service Directory Cloud Domains Private Service Connect 	<p>← Certificate details DELETE</p> <p>mcr1-12a7b631-61f2-4cd0-961b-9a310c20e8aa</p> <p>In use by k8s2-ts-ktptb5u7-tp4-oab-aake-66ztd693</p> <table border="1"> <tr> <td>Certificate Type</td> <td>MANAGED</td> </tr> <tr> <td>Status</td> <td>ACTIVE ?</td> </tr> <tr> <td>Domain status</td> <td> <ul style="list-style-type: none"> ✅ awi.<Your IP Address>.nip.io ✅ rest.<Your IP Address>.nip.io ✅ ws.<Your IP Address>.nip.io </td> </tr> <tr> <td>Expires</td> <td>Feb 7, 2022, 11:17:39 AM</td> </tr> <tr> <td>Serial Number</td> <td>26:1C:8B:48:37:21:6C:5F:09:00:00:00:00:CC:9C:70</td> </tr> <tr> <td>Certificate Issuer</td> <td>GTS CA 1D4</td> </tr> </table> <p>Certificate chain</p> <ul style="list-style-type: none"> GlobalSign Root CA - Jan 28, 2028, 1:00:42 AM GTS Root R1 - Jan 28, 2028, 1:00:42 AM GTS CA 1D4 - Sep 30, 2027, 2:00:42 AM awi.34.117.31.39.nip.io - Feb 7, 2022, 11:17:39 AM <p>Equivalent REST</p>	Certificate Type	MANAGED	Status	ACTIVE ?	Domain status	<ul style="list-style-type: none"> ✅ awi.<Your IP Address>.nip.io ✅ rest.<Your IP Address>.nip.io ✅ ws.<Your IP Address>.nip.io 	Expires	Feb 7, 2022, 11:17:39 AM	Serial Number	26:1C:8B:48:37:21:6C:5F:09:00:00:00:00:CC:9C:70	Certificate Issuer	GTS CA 1D4
Certificate Type	MANAGED												
Status	ACTIVE ?												
Domain status	<ul style="list-style-type: none"> ✅ awi.<Your IP Address>.nip.io ✅ rest.<Your IP Address>.nip.io ✅ ws.<Your IP Address>.nip.io 												
Expires	Feb 7, 2022, 11:17:39 AM												
Serial Number	26:1C:8B:48:37:21:6C:5F:09:00:00:00:00:CC:9C:70												
Certificate Issuer	GTS CA 1D4												

Chapter 7: Expose the Cluster to the outside world

To access the AWI and for agents to connect to the JCP, these services must be exposed via Ingresses and HTTP(S) Load Balancers.

The domains/endpoints where the HTTP(S) Load Balancers can be reached are configured as hosts in the Ingresses. The previously created TLS certificate can be referenced via the managed certificates extension.

When an Ingress object is created, the GKE Ingress controller automatically creates an HTTP(S) Load Balancer and configures it according to the information in the Ingress and the corresponding backend services.

The Ingress configuration to access AWI, JCP-REST and JCP-WS could look as below: Additionally, the endpoints for the WS and REST JCPs need to be configured in UC_SYSTEM_SETTINGS to also point to the Load Balancer address.

```
$ kubectl apply -f ingress.yaml
```

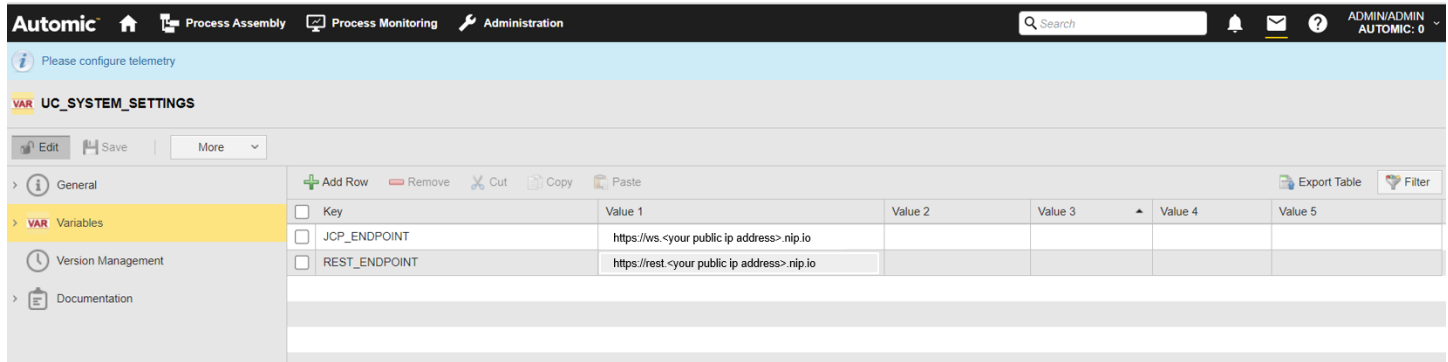
ingress.yaml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: aake-oab-ingress
  annotations:
    kubernetes.io/ingress.global-static-ip-name: oab-aake-static
    networking.gke.io/managed-certificates: oab-aake-cert-static
spec:
  rules:
    - host: awi.<your static public ip address>.nip.io
      http:
        paths:
          - path: /*
            pathType: ImplementationSpecific
            backend:
              service:
                name: awi
                port:
                  name: awi
    - host: rest.<your static public ip address>.nip.io
      http:
        paths:
          - path: /*
            pathType: ImplementationSpecific
            backend:
              service:
                name: jcp-rest
                port:
                  name: rest
    - host: ws.<your static public ip address>.nip.io
      http:
        paths:
          - path: /*
            pathType: ImplementationSpecific
            backend:
              service:
```

```
name: jcp-ws
port:
  name: ws
```

After the Ingresses have been successfully deployed and Load Balancers created, AWI can be reached via the exposed endpoint - `https://awi.<your static public ip address>.nip.io`

Additionally, the endpoints for the WS and REST JCPs need to be configured in UC_SYSTEM_SETTINGS to also point to the Load Balancer address.



If you already have domains/addresses assigned to the Load Balancer(s), you can also configure the endpoints as environment variables in values.yaml.

```
# environment defines variables that will be stored in the configmap aa-properties and injected as ENV into the containers
```

```
environment:
```

```
  JCP_WS_EXTERNAL_ENDPOINT: "https://ws-default.<your public ip address>.nip.io"
```

```
  JCP_REST_EXTERNAL_ENDPOINT: "https://rest-default.<your public ip address>.nip.io"
```

Chapter 8: Connect Agents via HTTPS Load Balancer

8.1 Connect TLS-enabled agents

TLS agents can connect to the JCPs via the Ingress/HTTPS Load Balancer that acts as a server for the TLS Handshake. The certificate of the LB needs to be trusted by the agent. Since a public CA signs it, it is usually trusted by applications since the root certificate of the signing CA is already included in the Java/OS truststore.

In this case, the ini file of the v21 Windows agent and TLS Gateway only require the Automic system name and endpoint where the JCPs can be reached:

UCXJWX6.ini:

```
[GLOBAL]
;
name=WINTLS01
;
system=AUTOMIC
...
[TCP/IP]
;
connection=ws-default.<your public ip address>.nip.io:443
```

8.2 Connect non-TLS agents via TLS Gateway

To use the TLS Gateway in CP mode, the TLS_GATEWAY_CP key in UC_SYSTEM_SETTINGS variable has to be set to Yes, and the cp_port ini parameter has to be configured.

Configure the required param in the ini file of the Gateways as below:

uctlsgtw.ini:

```
[GLOBAL]
;
name=TLSTW01
;
system=AUTOMIC
...
[TCP/IP]
;
connection=ws-default.<your public ip address>.nip.io:443
...
cp_port=2217
```

The v12.3 agents can use the same system name and the cp parameter has to match the hostname/address of the machine where the TLS Gateway is installed and also the same port configured as a cp_port for the Gateway.

For our example, this would be:

UCXJWX6.ini:

```
[GLOBAL]
;
name=WIN12.3
;
system=AUTOMIC
...
[TCP/IP]
;
cp=<your tls gateway hostname or address>:2217
```

ucxjsqlx.ini:

```
[GLOBAL]
;
name=SQL12.3
;
system=AUTOMIC
...
[TCP/IP]
;
cp=<your tls gateway hostname or address>:2218
```

The TLS Gateways and the agents should be visible in AWI.

