Installing Automic Automation Kubernetes Edition v21

How to deploy to AWS

Version 1.1

Contents

Chapter 1: Introduction

The Elastic Kubernetes Service (EKS) offered on AWS can be used to deploy and manage the Automic Automation Kubernetes Edition.

If you are new to AWS and Kubernetes, the EKS user guide can provide you with all the information you need, from installing and configuring the AWS CLI to the steps required to set up Fargate for your cluster. AWS provide a tutorial: Creating a VPC with Public and Private Subnets for Your Clusters to help with the correct network setup before creating the cluster.

Correctly configuring the IAM roles and policies can be a challenge. Still, there is plenty of documentation for this, and especially when using Fargate, don't forget to include the Core DNS pods and remove the ec2 annotations.

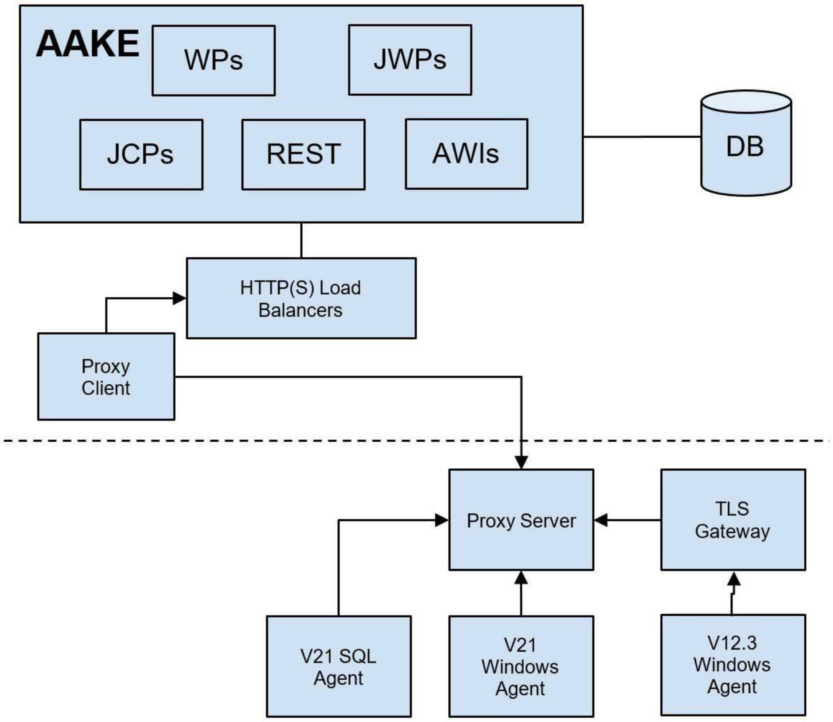
To expose the cluster to the outside world, you can deploy the AWS Load Balancer Controller to automatically create an application or network load balancer bound to Kubernetes services or Ingresses in the cluster.

AWS prerequisites at a glance:

- VPC with public and private subnetworks for Fargate
- EKS cluster with Fargate profiles (AAKE pods can use the default namespace)
- PostgreSQL RDS database instance
- AWS Load Balancer Controller

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 EKS. There are multiple options with the many services AWS provides, some of which might better fit your needs. In the scenario described in this guide, there is no need for CPs since all agents can connect to the new JCPs by taking advantage of the TLS Gateway. If using strict Firewall rules between security zones that only allow outgoing connections, the Automic Proxy pair can establish connections between the AAKE cluster and other components. An overview can be found in the diagram below.



Chapter 2: Create the PostgreSQL Database for Automic

There are some mandatory settings to optimize the database and connections for the Automic Server.

You will need to create a new parameter group and set vacuum_cost_limit to 10000 and client_encoding to LATIN9.

You can do this during the creation of the database instance or by modifying the configuration afterwards.

Once the instance is available, you can connect to it and create a new database:

```
$ psql --host=automic-db.id.eu-central-1.rds.amazonaws.com --port=5432 \
       --username=oab --password --dbname=postgres
```

```
  postgres=> CREATE DATABASE ae WITH OWNER = "oab" TEMPLATE = template0 ENCODING =
'UTF8' LC_COLLATE = 'C' LC_CTYPE = 'C' CONNECTION LIMIT = -1;
  postgres=> \c ae
  ae=> CREATE SCHEMA dbo AUTHORIZATION "oab";
  ae=> ALTER ROLE "oab" IN DATABASE ae SET search_path TO 'dbo';
```

Note: No additional tablespaces were created in this example, so the PostgreSQL default of pg_default will be used.

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=automic-db.id.eu-
central-1.rds.amazonaws.com \  --from-literal=vendor=postgres --from-
literal=port='5432' --from-literal=user=oab \ --from-literal=db=ae --from-
literal=password=automic \ --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

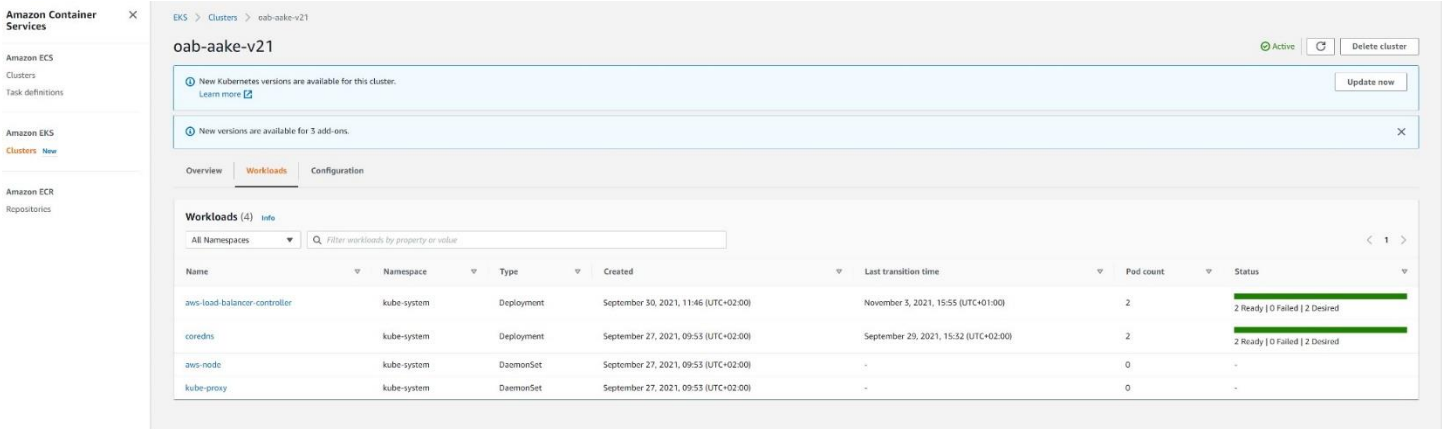
```
$ kubectl create secret generic client0-user --from-literal=client='0' \ --from-
literal=user=ADMIN --from-literal=department=ADMIN --from-literal=password=admin
```

Chapter 4: Install the AWS Load Balancer Controller

In this guide, the AWS Load Balancer Controller automatically provides Application Load Balancers when Ingress objects are created. You can find detailed instructions in the Amazon EKS User Guide.

You need to set the region and vpcid explicity to use the Controller with Fargate.

```
$ helm upgrade -i aws-load-balancer-controller eks/aws-load-balancer-controller \
  --set clusterName=oab-aake-v21 \
  --set serviceAccount.create=false \
  --set serviceAccount.name=automic-aws-load-balancer-controller \
  --set region=eu-central-1 \
  --set vpcId=<your vpc id> \
  -n kube-system
```



oab-aake-v21

New Kubernetes versions are available for this cluster.
Learn more

New versions are available for 3 add-ons.

Workloads (4) Info

Name	Namespace	Type	Created	Last transition time	Pod count	Status
aws-load-balancer-controller	kube-system	Deployment	September 30, 2021, 11:46 (UTC+02:00)	November 3, 2021, 15:55 (UTC+01:00)	2	2 Ready | 0 Failed | 2 Desired
coredns	kube-system	Deployment	September 27, 2021, 09:53 (UTC+02:00)	September 29, 2021, 15:32 (UTC+02:00)	2	2 Ready | 0 Failed | 2 Desired
aws-node	kube-system	DaemonSet	September 27, 2021, 09:53 (UTC+02:00)	-	0	-
kube-proxy	kube-system	DaemonSet	September 27, 2021, 09:53 (UTC+02:00)	-	0	-

Chapter 5: Deploy AAKE in the EKS 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 Adapt the resources for awi, jwp, jcp-ws and jcp-rest in values.yaml

Fargate creates nodes as needed during deployment; some pods might require more resources than allocated by default. Setting values for resource requests and limits ensures that the nodes are sized according to the pods' needs.

```
jwp:
  # resources used for pods of deployment: requests and limits, recommended memory
range = 700MB-2GB
  resources:
    requests:
      memory: "700Mi"
      cpu: "250m"
    limits:
      memory: "2Gi"
      cpu: "500m"
```

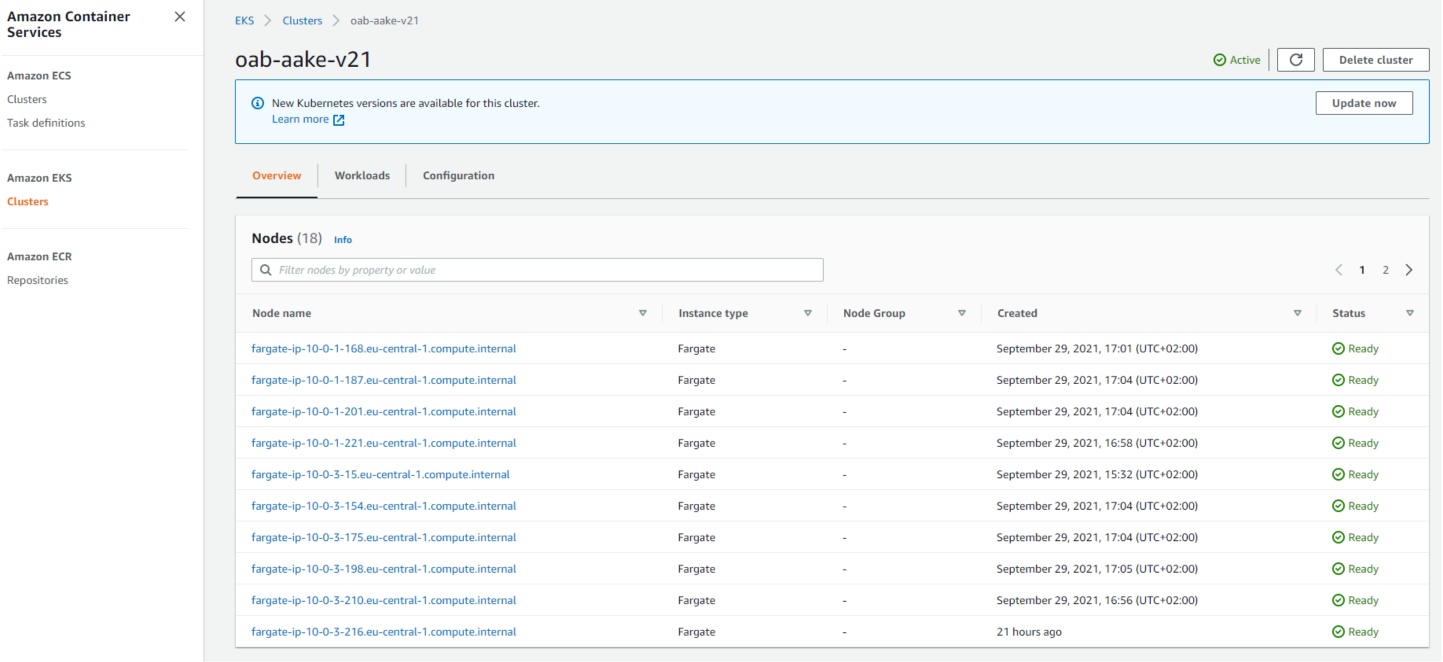
5.3 Configure the AWS CLI to connect to the cluster via command line

```
$ aws eks update-kubeconfig --region eu-central-1 --name oab-aake-v21
```

5.4 Install AAKE using Helm

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

The Fargate nodes are created on demand during the deployment as visualized below:



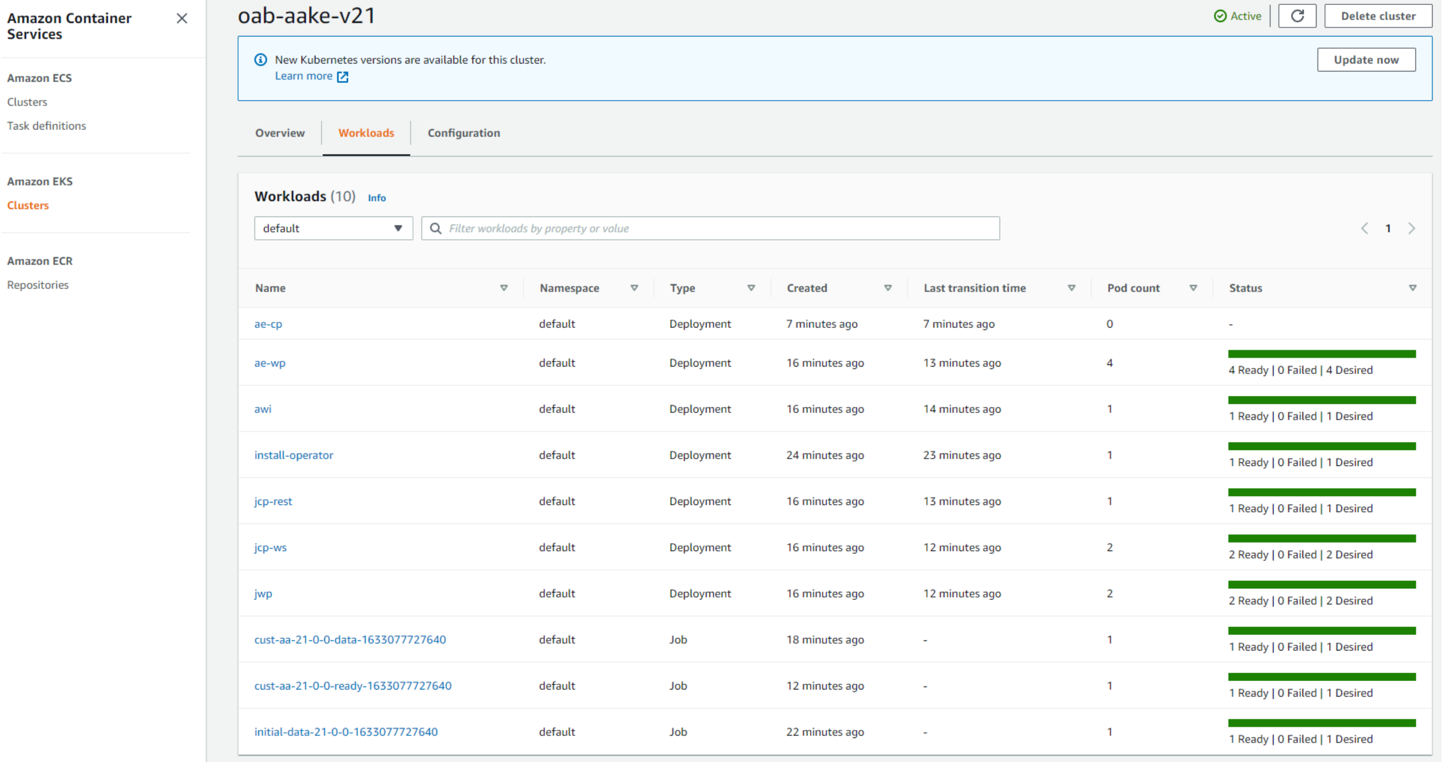
oab-aake-v21

New Kubernetes versions are available for this cluster.
Learn more

Nodes (18)

Node name	Instance type	Node Group	Created	Status
fargate-ip-10-0-1-168.eu-central-1.compute.internal	Fargate	-	September 29, 2021, 17:01 (UTC+02:00)	Ready
fargate-ip-10-0-1-187.eu-central-1.compute.internal	Fargate	-	September 29, 2021, 17:04 (UTC+02:00)	Ready
fargate-ip-10-0-1-201.eu-central-1.compute.internal	Fargate	-	September 29, 2021, 17:04 (UTC+02:00)	Ready
fargate-ip-10-0-1-221.eu-central-1.compute.internal	Fargate	-	September 29, 2021, 16:58 (UTC+02:00)	Ready
fargate-ip-10-0-3-15.eu-central-1.compute.internal	Fargate	-	September 29, 2021, 15:32 (UTC+02:00)	Ready
fargate-ip-10-0-3-154.eu-central-1.compute.internal	Fargate	-	September 29, 2021, 17:04 (UTC+02:00)	Ready
fargate-ip-10-0-3-175.eu-central-1.compute.internal	Fargate	-	September 29, 2021, 17:04 (UTC+02:00)	Ready
fargate-ip-10-0-3-198.eu-central-1.compute.internal	Fargate	-	September 29, 2021, 17:05 (UTC+02:00)	Ready
fargate-ip-10-0-3-210.eu-central-1.compute.internal	Fargate	-	September 29, 2021, 16:56 (UTC+02:00)	Ready
fargate-ip-10-0-3-216.eu-central-1.compute.internal	Fargate	-	21 hours ago	Ready

On the Workloads page it is possible to view the Automic deployments with the configured replicaset



oab-aake-v21

New Kubernetes versions are available for this cluster.
Learn more

Workloads (10)

Name	Namespace	Type	Created	Last transition time	Pod count	Status
ae-cp	default	Deployment	7 minutes ago	7 minutes ago	0	-
ae-wp	default	Deployment	16 minutes ago	13 minutes ago	4	4 Ready | 0 Failed | 4 Desired
awi	default	Deployment	16 minutes ago	14 minutes ago	1	1 Ready | 0 Failed | 1 Desired
install-operator	default	Deployment	24 minutes ago	23 minutes ago	1	1 Ready | 0 Failed | 1 Desired
jcp-rest	default	Deployment	16 minutes ago	13 minutes ago	1	1 Ready | 0 Failed | 1 Desired
jcp-ws	default	Deployment	16 minutes ago	12 minutes ago	2	2 Ready | 0 Failed | 2 Desired
jwp	default	Deployment	16 minutes ago	12 minutes ago	2	2 Ready | 0 Failed | 2 Desired
cust-aa-21-0-0-data-1633077727640	default	Job	18 minutes ago	-	1	1 Ready | 0 Failed | 1 Desired
cust-aa-21-0-0-ready-1633077727640	default	Job	12 minutes ago	-	1	1 Ready | 0 Failed | 1 Desired
initial-data-21-0-0-1633077727640	default	Job	22 minutes ago	-	1	1 Ready | 0 Failed | 1 Desired

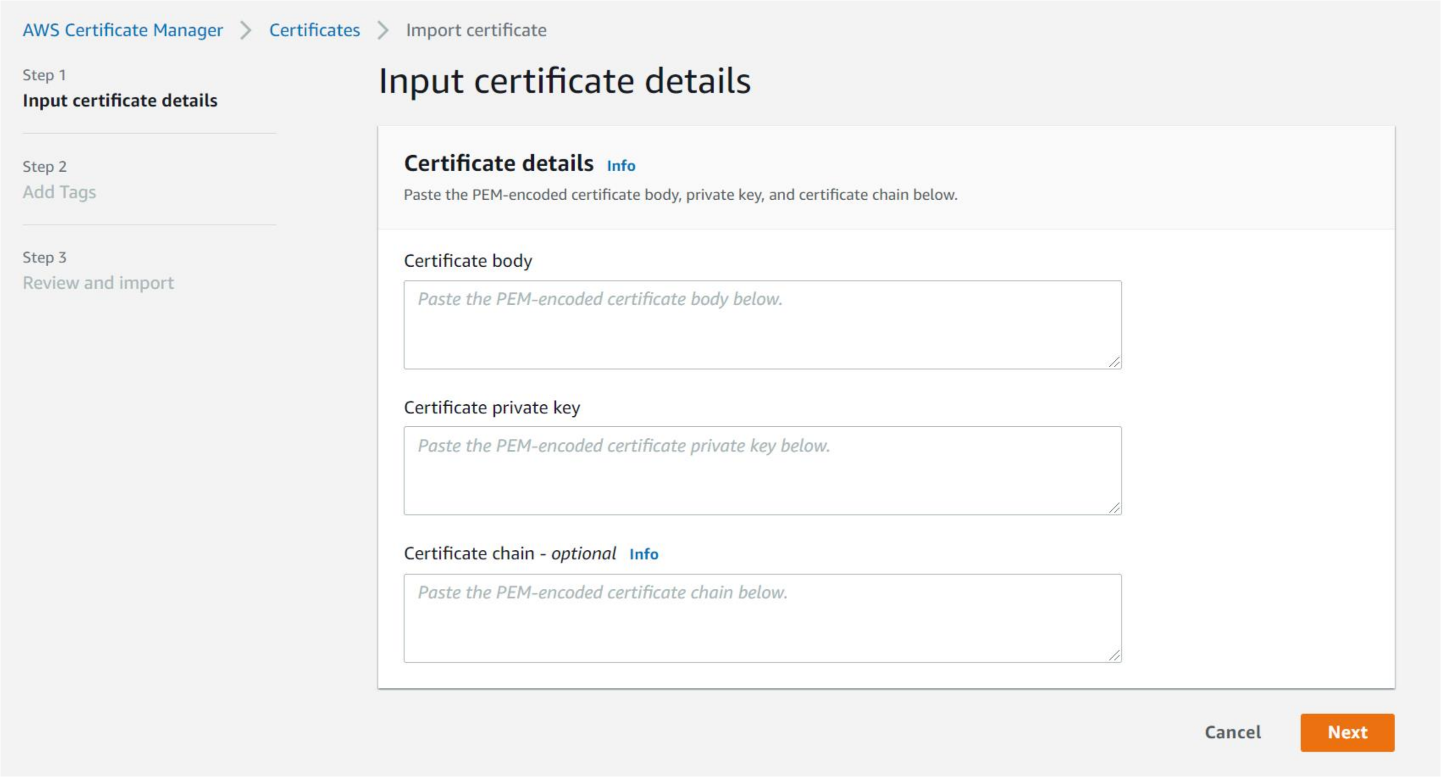
The Automic Helm plugin can be used to check the status of the installation:

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

Chapter 6: Configure TLS certificates in AWS Certificate Manager

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

The AWS Certificate Manager allows you to request a new certificate, but it is also possible to import an existing private key and certificate, as is the case in this guide.



AWS Certificate Manager > Certificates > Import certificate
Step 1
Input certificate details
Step 2
Add Tags
Step 3
Review and import
Info
Info
Cancel
Next

Input certificate details

Certificate details

Paste the PEM-encoded certificate body, private key, and certificate chain below.

Certificate body

Paste the PEM-encoded certificate body below.

Certificate private key

Paste the PEM-encoded certificate private key below.

Certificate chain - *optional*

Paste the PEM-encoded certificate chain below.

Ensure to include the domains of all the HTTPS Load Balancers in the certificate if you don’t use a wildcard domain; otherwise, the agents will not connect because of the TLS hostname verification.

Chapter 7: Expose the Cluster to the outside world

To access the AWI and for TLS agents/Gateway to connect to the JCP, you have to expose the cluster services via Ingresses and HTTP(S) Load Balancers.

The domains/endpoints for the HTTP(S) Load Balancers are configured as hosts in the Ingresses, and the TLS certificate is referenced via the AWS Certificate Manager ARN.

With Fargate, it’s only possible to use IP targets to register pods as targets for Load Balancers.

The Ingress configuration to access AWI, JCP-REST and JCP-WS via 3 HTTP(S) Load Balancers could look as below:

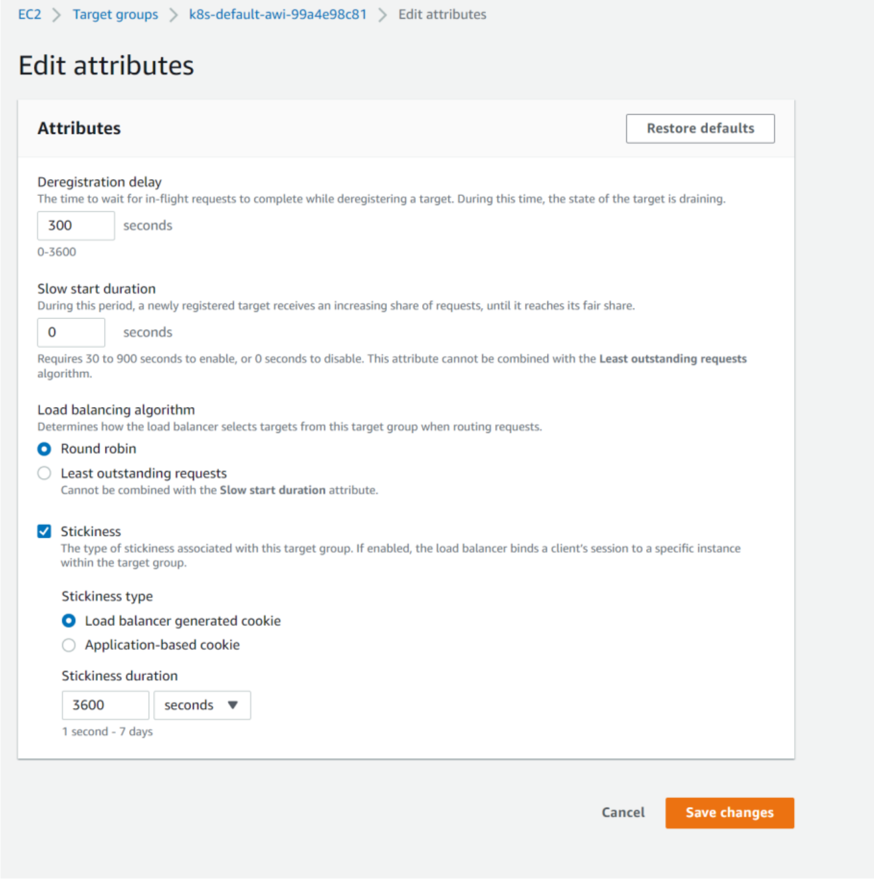
```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: aake-awi
  annotations:
    kubernetes.io/ingress.class: alb
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTPS":443}, {"HTTP":80}]'
spec:
  rules:
    - host: <your awi domain name>
      http:
        paths:
          - path: /*
            backend:
              serviceName: awi
              servicePort: awi


apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: aake-rest
  annotations:
    kubernetes.io/ingress.class: alb
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTPS":443}, {"HTTP":8088},
{"HTTP": 80}]'
spec:
  rules:
    - host: <your jcp rest domain name>
      http:
        paths:
          - path: /*
            backend:
              serviceName: jcp-rest
              servicePort: rest


apiVersion: networking.k8s.io/v1beta1
```

```
kind: Ingress
metadata:
  name: aake-ws
  annotations:
    kubernetes.io/ingress.class: alb
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTPS": 443}, {"HTTPS":8443}]'
    alb.ingress.kubernetes.io/backend-protocol: HTTPS
    alb.ingress.kubernetes.io/certificate-arn: arn:aws:acm:eu-central-
1:564875751664:certificate/b4ae0dfe-baf6-4939-b0c2-736e33823500
spec:
  rules:
    - host: <your jcp ws domain name>
      http:
        paths:
          - path: /*
            backend:
              serviceName: jcp-ws
              servicePort: ws
```

When AWI is used with Load Balancers, sticky sessions are required and they can be configured by editing the EC2 Target Group for AWI:


EC2 > Target groups > k8s-default-awi-99a4e98c81 > Edit attributes

Edit attributes

Attributes

Deregistration delay
The time to wait for in-flight requests to complete while deregistering a target. During this time, the state of the target is draining.
300 seconds
0-3600

Slow start duration
During this period, a newly registered target receives an increasing share of requests, until it reaches its fair share.
0 seconds
Requires 30 to 900 seconds to enable, or 0 seconds to disable. This attribute cannot be combined with the **Least outstanding requests** algorithm.

Load balancing algorithm
Determines how the load balancer selects targets from this target group when routing requests.
● Round robin
○ Least outstanding requests
Cannot be combined with the **Slow start duration** attribute.

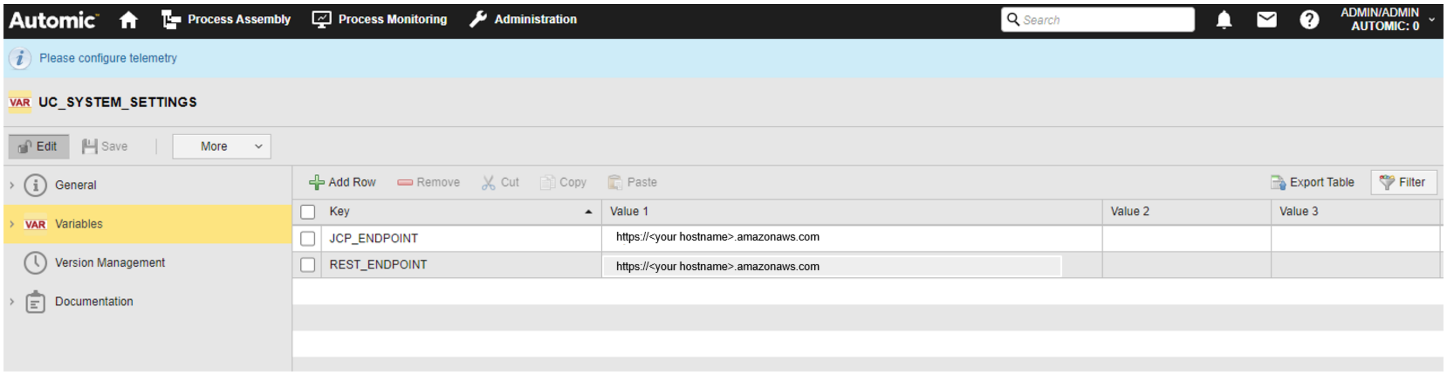
☑ Stickiness
The type of stickiness associated with this target group. If enabled, the load balancer binds a client's session to a specific instance within the target group.

Stickiness type
● Load balancer generated cookie
○ Application-based cookie

Stickiness duration
3600 seconds ▼
1 second - 7 days

After the Ingresses have been successfully deployed and Load Balancers created, AWI can be reached via the exposed endpoint
https://<your awi domain name>.

Additionally, the WS and REST JCP endpoints need to be configured in UC_SYSTEM_SETTINGS to also point to the Load Balancer domains.



Key	Value 1	Value 2	Value 3
JCP_ENDPOINT	https://<your hostname>.amazonaws.com		
REST_ENDPOINT	https://<your hostname>.amazonaws.com		

If you already have domains 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://<your jcp ws domain name>"
  JCP_REST_EXTERNAL_ENDPOINT: "https://<your jcp rest domain name>"
```

Chapter 8: Connect agents via HTTPS Load Balancer

8.1 Install the Automic Proxy

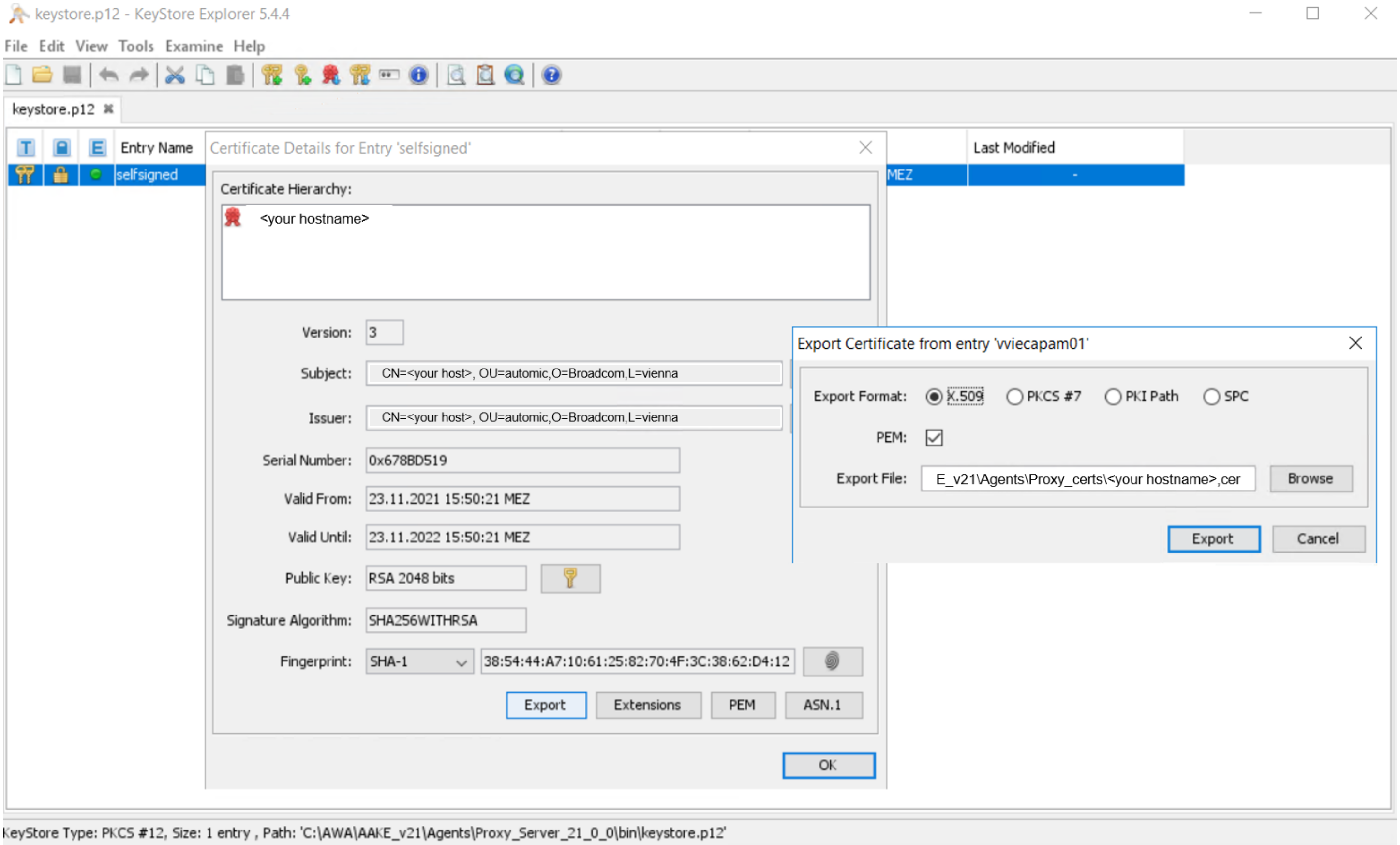
You can follow the steps in the Automic documentation to install the 2 Proxy components https://docs.automic.com/documentation/webhelp/english/AA/21.0/DOCU/21.0/Automic%20Automation%20Guides/Content/Proxy/installation_proxy.htm?Highlight=proxy.

The Proxy Client will connect to the JCPs via the Ingress/HTTPS Load Balancer, while the Proxy Server can accept connections from the TLS Gateway and TLS-enabled agents.

The Ingress/Load Balancer(s) server certificate that is configured in AWS Certificate Manager needs to be trusted by the Proxy Client.

If you use a self-signed certificate or one signed by an internal CA, you can import this into the Java truststore of the JRE used to start the Proxy Client.

A self-signed certificate has to be generated before starting the Proxy Server as it works with a Java keystore. Self-signed certificate can be exported from the keystore with KeyStore Explorer or a similar tool.


keystore.p12 - KeyStore Explorer 5.4.4
File Edit View Tools Examine Help
keystore.p12
T
E
Entry Name
Last Modified
selfsigned
MEZ
-
Certificate Details for Entry 'selfsigned'
Certificate Hierarchy:
<your hostname>
Export
Extensions
PEM
ASN.1
OK
Export Certificate from entry 'vviecapam01'
Browse
Export
Cancel

Version: 3

Subject: CN=<your host>, OU=automic,O=Broadcom,L=vienna

Issuer: CN=<your host>, OU=automic,O=Broadcom,L=vienna

Serial Number: 0x678BD519

Valid From: 23.11.2021 15:50:21 MEZ

Valid Until: 23.11.2022 15:50:21 MEZ

Public Key: RSA 2048 bits

Signature Algorithm: SHA256WITHRSA

Fingerprint: SHA-1 38:54:44:A7:10:61:25:82:70:4F:3C:38:62:D4:12

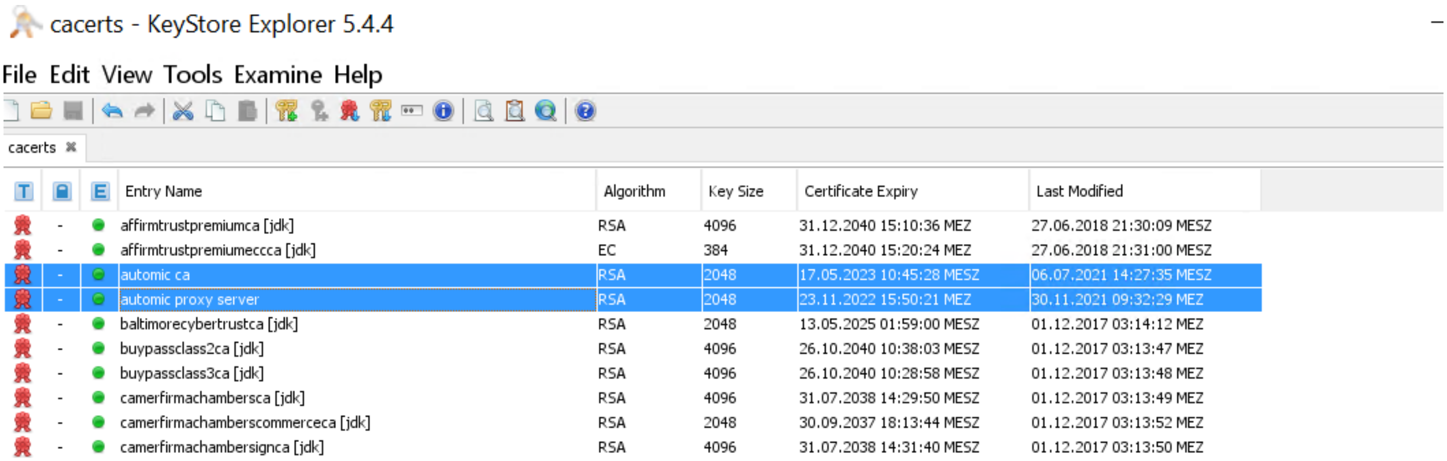
Export Format: ● X.509 ○ PKCS #7 ○ PKI Path ○ SPC

PEM: ☑

Export File: E_v21\Agents\Proxy_certs\<your hostname>,cer

KeyStore Type: PKCS #12, Size: 1 entry , Path: 'C:\AWA\AAKE_v21\Agents\Proxy_Server_21_0_0\bin\keystore.p12'

For the Proxy Server certificate to be trusted by the Proxy Client, it will be imported in the Java cacerts via the KeyStore Explorer (running as administrator might be required), the same as for the HTTPS Load Balancer certificate.



cacerts - KeyStore Explorer 5.4.4
File Edit View Tools Examine Help
cacerts

T	🔒	E	Entry Name	Algorithm	Key Size	Certificate Expiry	Last Modified
🏵	-	●	affirmtrustpremiumca [jdk]	RSA	4096	31.12.2040 15:10:36 MEZ	27.06.2018 21:30:09 MESZ
🏵	-	●	affirmtrustpremiumeccca [jdk]	EC	384	31.12.2040 15:20:24 MEZ	27.06.2018 21:31:00 MESZ
🏵	-	●	automic ca	RSA	2048	17.05.2023 10:45:28 MESZ	06.07.2021 14:27:35 MESZ
🏵	-	●	automic proxy server	RSA	2048	23.11.2022 15:50:21 MEZ	30.11.2021 09:32:29 MEZ
🏵	-	●	baltimorecybertrustca [jdk]	RSA	2048	13.05.2025 01:59:00 MESZ	01.12.2017 03:14:12 MEZ
🏵	-	●	buypassclass2ca [jdk]	RSA	4096	26.10.2040 10:38:03 MESZ	01.12.2017 03:13:47 MEZ
🏵	-	●	buypassclass3ca [jdk]	RSA	4096	26.10.2040 10:28:58 MESZ	01.12.2017 03:13:48 MEZ
🏵	-	●	camerfirmachambersca [jdk]	RSA	4096	31.07.2038 14:29:50 MESZ	01.12.2017 03:13:49 MEZ
🏵	-	●	camerfirmachamberscommerceca [jdk]	RSA	2048	30.09.2037 18:13:44 MESZ	01.12.2017 03:13:52 MEZ
🏵	-	●	camerfirmachambersignca [jdk]	RSA	4096	31.07.2038 14:31:40 MESZ	01.12.2017 03:13:50 MEZ

Proxy Client ini file can then be configured as below:

```
[GLOBAL]
;
name=PROXY01
;
system=AUTOMIC
;
serverProxy=<your server proxy hostname or address>:4321
;
routingPort=8445
...
[TCP/IP]
;
; connection: Connection Parameter: Address of the endpoint used to connect to the
AE system.
; Allowed formats:
; DNS Name:Port number
; TCP/IP Address:Port number
;
connection=<your jcp ws domain name>:8443
```

8.2 Connect TLS-enabled agents to the Proxy Server

The agents connecting to the Proxy Server need to trust its self-signed certificate, either by importing the Proxy Server certificate in the Java or OS truststore on the host where the agent runs or copying it to a folder accessible to the agent.

If the certificate is imported into the Java/OS truststore where the agent/TLS Gateway are installed, the ini file of the v21 Windows agent and TLS Gateway only require the Automic system name and hostname/address of the Proxy Server:

UCXJWX6.ini:

```
[GLOBAL]
;
name=WINTLS01
;
system=AUTOMIC
...
```

```
[TCP/IP]
;
connection=<your server proxy hostname or address>:8445
```

ucxjsqlx.ini:

```
[GLOBAL]
;
name=SQLTLS01
;
system=AUTOMIC
…
[TCP/IP]
;
connection=<your server proxy hostname or address>:8445
```

8.3 Connect non-TLS agents via the TLS Gateway and Proxy Server

To use the TLS Gateway in CP mode, the TLS_GATEWAY_CP key in the UC_SYSTEM_SETTINGS variable must be set to Yes, and the cp_port ini parameter has to be configured. The Gateway must trust the Proxy Server certificate, same as for the TLS-enabled agents. The required parameter in the ini file of the Gateway can be configured as below:

uctlsgtw.ini:

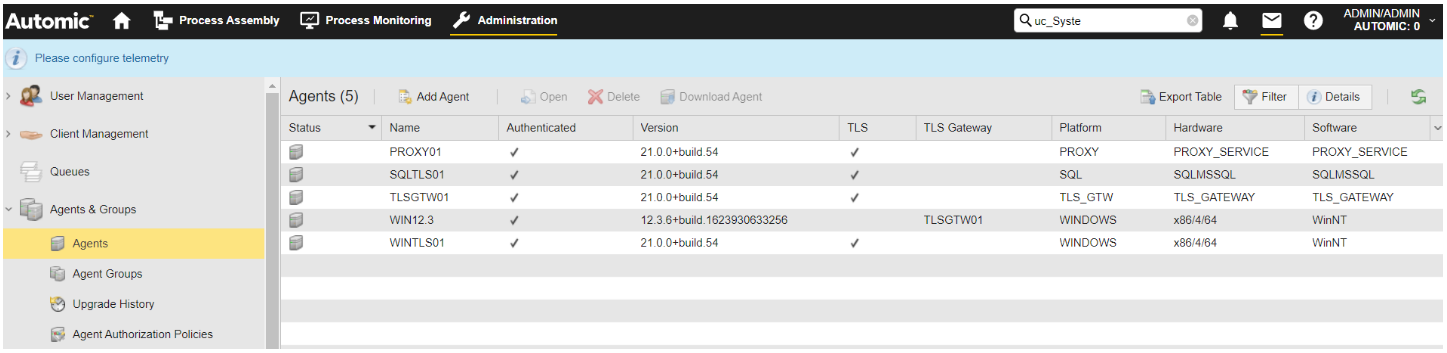
```
[GLOBAL]
;
name=TLSGTW01
;
system=AUTOMIC
…
[TCP/IP]
;
connection=<your server proxy hostname or address>:8445
…
cp_port=2217
```

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.

UCXJWX6.ini:

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

The Automic Proxy, TLS Gateway and the agents should be visible in AWI



Status	Name	Authenticated	Version	TLS	TLS Gateway	Platform	Hardware	Software
	PROXY01	✓	21.0.0+build.54	✓		PROXY	PROXY_SERVICE	PROXY_SERVICE
	SQLTLS01	✓	21.0.0+build.54	✓		SQL	SQLMSSQL	SQLMSSQL
	TLSGTW01	✓	21.0.0+build.54	✓		TLS_GTW	TLS_GATEWAY	TLS_GATEWAY
	WIN12.3	✓	12.3.6+build.1623930633256		TLSGTW01	WINDOWS	x86/4/64	WinNT
	WINTLS01	✓	21.0.0+build.54	✓		WINDOWS	x86/4/64	WinNT

BROADCOM®