



Provision AWS Services through Tanzu Application Service Using the AWS Service Broker for VMware Tanzu

by Ryan Niksch, Senior Solution Architect AWS

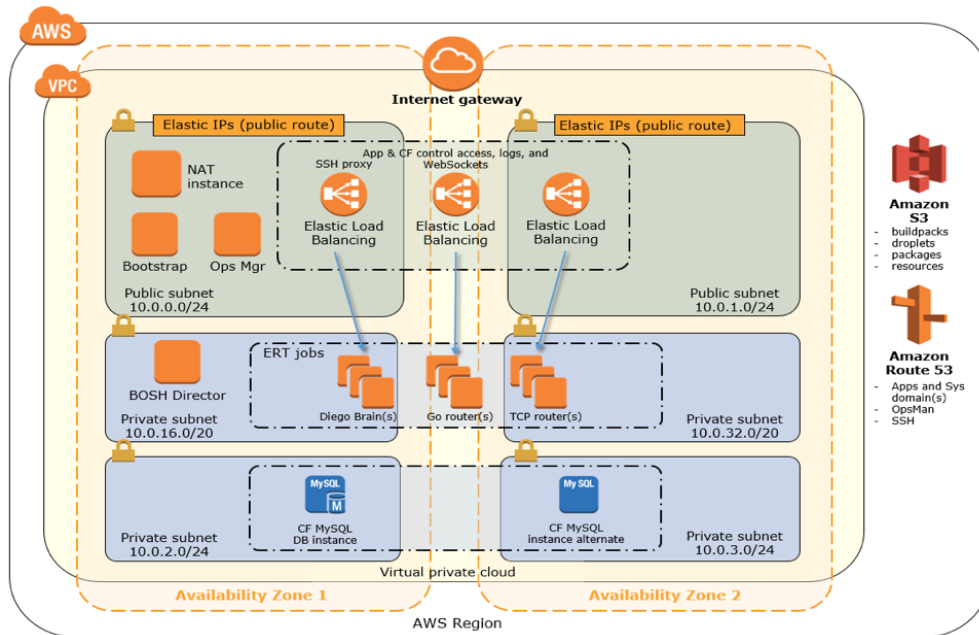
Enterprise customers tell us that they're looking for application modernization and greater agility. They may also need to work in a hybrid context, and want technology that makes it easy to run in multiple environments now, and in the future.

To meet these needs, they are choosing application platforms such as Tanzu Application Service (and the upstream open source Cloud Foundry). Yes, there are many do-it-yourself solutions in the market, but these require investments of resources and effort to integrate, and in some cases custom security and compliance components must be developed as well. With Tanzu Application Service, you can instead focus on building your business-critical apps, on a platform that also provides "guard rails" and a rich feature set.

There are two open source projects that you can use to get a Tanzu Application Service production solution up and running on AWS. Once there, you can accelerate the integration and adoption of native AWS services such as Amazon Relational Database Service ([Amazon RDS](#)), [Amazon DynamoDB](#), Amazon Simple Queue Service ([SQS](#)) and Amazon Simple Storage Service ([Amazon S3](#)) into your applications stacks, as part of your modernization process.

In this post, I'll first show you how to rapidly provision Tanzu Application Service on AWS through an AWS Quick Start. Then I'll demonstrate using the [AWS Service Broker](#) – now available via [Tanzu Network](#) – to gain further agility and modernize your application stacks. Customer's look to the Tanzu Network as their trusted source for sourcing the binaries they rely on for operating, securing, and extending their platforms' capabilities.

The [Tanzu Application Service \(TAS\) on AWS Cloud Quick Start](#) is an open source CloudFormation-based reference deployment which can be provisioned in a few hours by filling in a few parameters. This solution will build out a production-ready, scalable, resilient, reference architecture (diagram below) which takes advantage of multiple availability zones (AZs), with a managed Amazon Relational Database Service ([Amazon RDS](#)) deployed in a multi-AZ configuration. Elastic Load Balancers allow for the addition of nodes without significant downtime.



The deployment guide will walk you through the process of creating the required Tanzu Network account and Route 53 Hosted DNS Zones. [AWS Certificate Manager \(ACM\)](#) makes requesting the required certificates very simple.

Once these three steps are complete, you will be ready to launch the CloudFormation stack which builds out Tanzu Application Service in your AWS account.

After Tanzu Application Service (TAS) is running in your AWS account, you'll be ready to integrate the AWS Service Broker for VMware Tanzu. With the Service Broker deployed, development teams can provision and expose native AWS services to their application workloads running in Cloud Foundry directly via the application platform. This could be TAS running applications on AWS via the Service Broker (catering for faster modernization and cloud services adoption), or applications running on TAS on-premises and consuming AWS services running in the cloud.

The Service Broker provides multi-account provisioning, allowing development teams to take advantage of account-level segregation for project and cost management. A common and recommended practice is to use separate accounts for development and production workloads. The Broker can be configured to provision to either, depending on use case.

Through the Broker, service plans build in best practices for high availability, encryption, and data retention. Customers can use configurable overrides to cater for specific use cases, and can customize the CloudFormation template in a private catalog to meet their specific business requirements.

The Service Broker provides a powerful, elegant way to interact with add-on services: with just a few commands, developers can bind their apps to over a dozen AWS services. This kind of convenience helps customers go faster and improve business outcomes with differentiated software.



Now let's get our hands dirty!

Tanzu Application Service on AWS Cloud Quick Start

First, follow the [Tanzu Application Service Quick Start Reference Deployment guide](#) to deploy the Pivotal Quick Start and Service Broker into the Oregon us-west-2 region.

Once that is complete, you can install and configure the AWS Service Broker in TAS. The broker uses a DynamoDB table as a persistent store for service instances, and as a distributed cache/lock. To create the table, run following command from the AWS CLI:

Bash

```
aws dynamodb create-table --attribute-definitions \  
AttributeDefinition={AttributeName=id,AttributeType=S} AttributeDefinition={AttributeName=userid,AttributeType=S} \  
AttributeDefinition={AttributeName=type,AttributeType=S} --key-scheme AttributeDefinition={AttributeName=id,KeyType=HASH} \  
AttributeDefinition={AttributeName=userid,KeyType=RANGE} --global-secondary-indexes \  
'IndexName=type-userid-index,KeySchema=[{AttributeName=type,KeyType=HASH},{AttributeName=userid,KeyType=RANGE}],Projection={ProjectionType=INCLUDE,NonKeyAttributes=[id,userid,type,locked]},ProvisionedThroughput={ReadCapacityUnits=5,WriteCapacityUnits=5}' \  
--provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5 \  
--region us-east-1 --table-name awssb
```

The Broker can be configured to use AWS Identity and Access Management ([IAM](#)) keys; however, we recommend making use of IAM roles, which can be attached to the cluster using an instance profile at launch.



The following is an example IAM policy to allow the Broker to interact with CloudFormation, DynamoDB, and other services. This policy should be attached to either an IAM user or role. If using a role, the role will need to be associated with the nodes with an instance profile.

```
Service User/Role
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "cloudformation:*",
        "ssm:*",
        "dynamodb:*",
        "s3:*"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/AWSServiceBrokerCFNRole"
      ],
      "Effect": "Allow"
    }
  ]
}
```

The Service Broker use a CloudFormation service role, which will supply the required permission to the Broker to provision and manage the AWS services provided through the service plans. **Caveat:** At the time of writing, the Broker uses a CloudFormation Service role. After an update expected later this year, it will instead use the IAM User or Role shown above. Check the Service Broker documentation for the current status.

In either case, the Service Broker has the permission needed to provision and control resources instead of providing access to specific teams. You can control which teams are able to interact with which Service Broker service plans via the Cloud Foundry role-based access controls.

Below is an example of a broad IAM policy which would enable *all* current service plans. This can be scoped down if only specific services are required:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
```



```
        "cloudformation:*",
        "iam:*",
        "kms:*",
        "ssm:*",
        "ec2:*",
        "lambda:*",
        "athena:*",
        "dynamodb:*",
        "elasticache:*",
        "elasticmapreduce:*",
        "rds:*",
        "redshift:*",
        "route53:*",
        "s3:*",
        "sns:*",
        "sqs:*",
        "polly:*",
        "lex:*",
        "translate:*",
        "rekognition:*",
        "kinesis:*"
    ],
    "Resource": [
        "*"
    ],
    "Effect": "Allow"
}
]
```

Install the AWS Service Broker for VMware Tanzu

Once the roles and DynamoDB table have been created, we can download and install the Broker. The AWS Service Broker now being listed on Tanzu Network is greatly beneficial For Customers who have corporate policies and security requirements stating that add-ons to Cloud Foundry foundations have to come from a trusted validated sources such as Tanzu Network.

In this example, I will be using the latest version of the AWS Service Broker for VMware Tanzu. First we have to go login into the Tanzu Network. Search for the AWS Service Broker for VMware Tanzu.



Explore, download, and update software and services

Q AWS Ser|



AWS Service Broker for VMware Tanzu

PARTNER SERVICES

VMware Tanzu

A unified, multi-cloud product to run your enterprise apps.



App Metrics



BOSH Backup and Restore

At the time of writing this Release 1.0.2 was available. Click on the download button on the bottom left of the screen.



AWS Service Broker for VMware Tanzu

[Documentation](#)

GET EMAIL ALERTS

Releases: 1.0.2

AWS Service Broker 1.0.2
15.1 MB 1.0.2

Release Details

RELEASE DATE	2020-07-15
RELEASE TYPE	Major Release
END OF GENERAL SUPPORT	2021-04-30

RELEASE DESCRIPTION
Initial release on Tanzu Network

DEPENDS ON

Products in the "Depends On" section must be installed prior to installing or upgrading to AWS Service Broker for VMware Tanzu 1.0.2. Please install or upgrade these products to one of the listed versions.

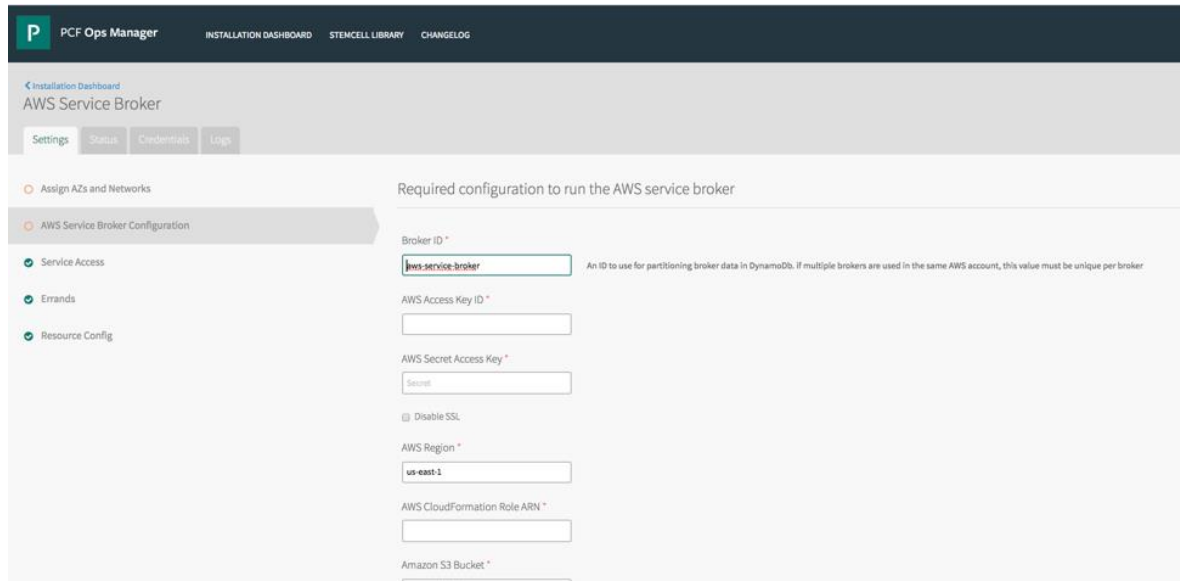
VMware Tanzu Application Service for VMs
[2.9*](#), [2.8*](#), [2.7*](#) or [2.6*](#)

Once downloaded the tile can be installed via the Tanzu Application Service Ops manager console as follows:

1. Connect to ops manager <https://opsman.<YOUR CF DOMAIN >>,
2. Click on **Import Product** on the top left.

3. Browse to the downloaded Service Broker tile.
4. The new tile install will appear on the left.
5. Click the + symbol to install the tile.

Once the Service Broker is installed, there are a few configuration steps which need to be completed. Click on the Service Broker tile, and select **Settings**.



If you are using an ec2 instance role attached to the Broker hosts, specify “use-role” as the value for both the AWS Access Key ID and AWS Secret Access. Otherwise, specify the credentials for the user. In this example, we’ll create an IAM user and attach the policy described above, then provide the Key ID and access key for that user in this configuration step.

Fill in the configuration parameters:

- AWS Region: this is the default region for the broker to deploy services into, and must match the region that the DynamoDB table created above (this will be decoupled in an upcoming update).
- AWS CloudFormation Role ARN: specify the ARN for the CloudFormation Role.
- Amazon S3 Bucket: specify awsservicebrokeralpha
- Amazon S3 Key Prefix: specify pcf/templates/
- Amazon S3 Region: specify us-west-2
- Amazon S3 Key Suffix: specify -main.yaml
- Amazon DynamoDB table name: specify awssb

Click **Save**.

Now **Apply Changes**. Once complete, the tile will change to green.



At this point, we have deployed Tanzu Application Service into an AWS account, and installed and configured the Service Broker.

Example Application

Now let's deploy a simple example application integrated with the Broker. This application will consume [Amazon Polly](#), provisioned through the AWS Service Broker.

Download the CF CLI. I am using an AWS Cloud9 developer IDE, so I will be adding the YUM repo to my list of repositories:

```
Bash
sudo wget -O /etc/yum.repos.d/cloudfoundry-cli.repo
https://packages.cloudfoundry.org/fedora/cloudfoundry-cli.repo
sudo yum install cf-cli
```

Clone the AWS Service Broker Polly Application:

```
Bash
git clone https://github.com/rniksch/aws-service-broker-polly-sample.git
```

Change dir into the sample app folder:

```
Bash
cd aws-service-broker-polly-sample/
```

Login to the Apps Manager using the CF CLI:

```
Bash
cf login -a https://api.sys.<YOUR CLOUD FOUNDRY DOMAIN> -u admin \
--skip-ssl-validation
```

Please note that this may not be the same password as the admin for the Cloud Foundry ops manager. If you are not sure of the password, you can collect it through this process:

1. Log in into the ops manager console:
2. <https://opsman.<YOUR CLOUD FOUNDRY DOMAIN>>
3. Select the **Tanzu Application Service** tile,
4. Select the **Credentials** tab,
5. Scroll down to **UAA Admin Credentials**,
6. Click on the link to your credential.
7. This will load a JSON payload with the password in it.

For this example, I have an existing Org called Blog and space called Polly.

First we will deploy one of the Service Broker plans, the Polly service.



Here's how to get a list of all available Service plans:

```
cf marketplace
```

The following command will deploy the AWS polly service via the Service Broker:

```
Bash
    cf create-service polly default blogpolly
```

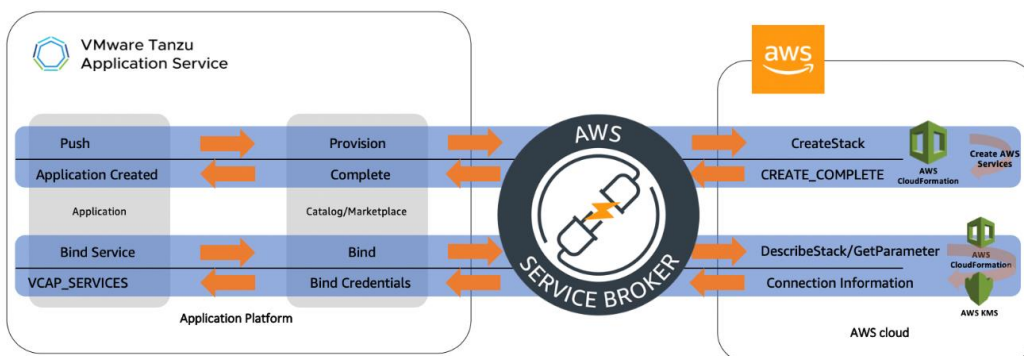
While the Service is deploying, let's push the sample application:

```
Bash
    cd aws-service-broker-polly-sample/
    cf push
```

Connecting to the URL of the pushed application will fail – the application is told to connect to Amazon Polly, but at this stage there is no binding between the Amazon Polly service we have created and the application.

So let's create that binding:

Provision and Bind



```
Bash
    cf bind-service pollysample blogpolly
```

We will need to restart the application for this to take effect:

```
Bash
    cf restage pollysample.
```



Get the URL for the application and test if it is working as expected:

```
Bash
  cf app pollysample
```

Copy the URL shown next to Routes into a browser.

This should load a black screen, with a play button that will convert the text in the app to audio via the AWS Polly Service.

It is important to note that at *no* point did the developer have to wait for a cloud or ops team to deploy the AWS service and provide connection details such as endpoint or credentials, at *no* point was the developer required to leave the developer interface and switch context to an AWS web console, and *no* in-depth understanding of the service was required.

Using the AWS Tanzu Application Quick Start and the AWS Service Broker, you can accelerate your deployment of Cloud Foundry into your AWS accounts, and build scalable, agile, elastic applications consuming native cloud services – more quickly, with less effort. Don't just modernize, enjoy the ride with the [AWS Service Broker](#)!

We look forward to seeing you contribute to the [AWS Service Broker](#) and [Quick Start](#) projects.



Ryan Niksch

Ryan Niksch is a Partner Solutions Architect focusing on application platforms, hybrid application solutions, and modernization. Ryan has worn many hats in his life and has a passion for tinkering and a desire to leave everything he touches a little better than when he found it.