Publishing to SAR

In this application, you'll learn how to publish your application to SAR.

WE'LL COVER THE FOLLOWING

- Adding the `Metadata` section
- Adding `README.md` and `LICENSE.md`
- Deployment Bucket
  - Using SAR or just sharing templates

Adding the `Metadata` section

In addition to making it easy to use SAR components in your applications, SAM makes it very easy to publish an application directly to the SAR repository. Now, you'll publish your work so far as a private application. You just need to add a bit of metadata about the application to the template. You can create a new top-level section called `Metadata` in `template.yaml`, with the code similar to the following listing (feel free to change the names).

```
Metadata:
  AWS::ServerlessRepo::Application:
    Name: image-thumbnails
    Description: >
      A sample application for the Running Serverless book tutorial
    Author: Gojko Adzic
    SemanticVersion: 1.0.0
    SpdxLicenseId: MIT
    LicenseUrl: LICENSE.md
    ReadmeUrl: README.md
    Labels: ['layer', 'image', 'lambda', 'imagemagick']
    HomePageUrl: https://runningserverless.com
    SourceCodeUrl: https://runningserverless.com
```

Line 136 to Line 148 of code/ch10/template.yaml

Most parameters have obvious purposes. Here is a bit more information about the less obvious parameters:

- `SemanticVersion` (line 7) is the version of the application. SAR allows you to publish multiple versions of a template, and clients can choose which version to install. Remember that you listed the application ID and the version when including the ImageMagick layer earlier in this chapter.
- `SpdxLicenseId` (line 8) is the Software Package Data Exchange (SPDX) identifier for the application copyright licence. When uploading private applications you do not need to include a licence, but this is necessary for public applications.
- `LicenseUrl` (line 9) is also optional for private applications and required for public ones. It points to a file in the function package repository with more information about the usage licence. It will be added in just a moment.
- `ReadmeUrl` (line 10) should point to a local file that contains the basic usage instructions for the application. It is also optional for private applications.

Adding `README.md` and `LICENSE.md`

Here, you'll create some additional files. You can use markdown to create basic markup such as headers and links. You can add a new file to the application directory containing `template.yaml`, called `README.md`, containing a description such as the following:

```
A sample application for the Running Serverless book tutorial
```

You also need a license file. Let's keep it simple. You can add a file called `LICENSE.md` with the following content.

```
MIT license
```

Deployment Bucket

Before you publish the application to the repository, you need to allow SAR to read templates from your deployment bucket. Create a file called `bucket-policy.json` with the content from the following listing (replace `BUCKET_NAME`

in line 10 in the final widget with the bucket you use to package SAM applications).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service":  "serverlessrepo.amazonaws.com"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::BUCKET_NAME/*"
        }
    ]
}
```

code/ch10/bucket-policy.json

You can upload this access policy for your bucket to AWS using the following command:

```
aws s3api put-bucket-policy --bucket BUCKET_NAME --policy fileb://bucket-policy.json
```

Note that you need to do this just once, and SAR will be able to read your bucket in the future. Now you can build and package the application, but you won't deploy it. Instead, you'll use the following command to publish it to SAR:

```
sam publish -t output.yaml
```

Don't forget to replace `BUCKET_NAME` in line 10 of the file `bucket-policy.json` in the widget below:

Environment Variables

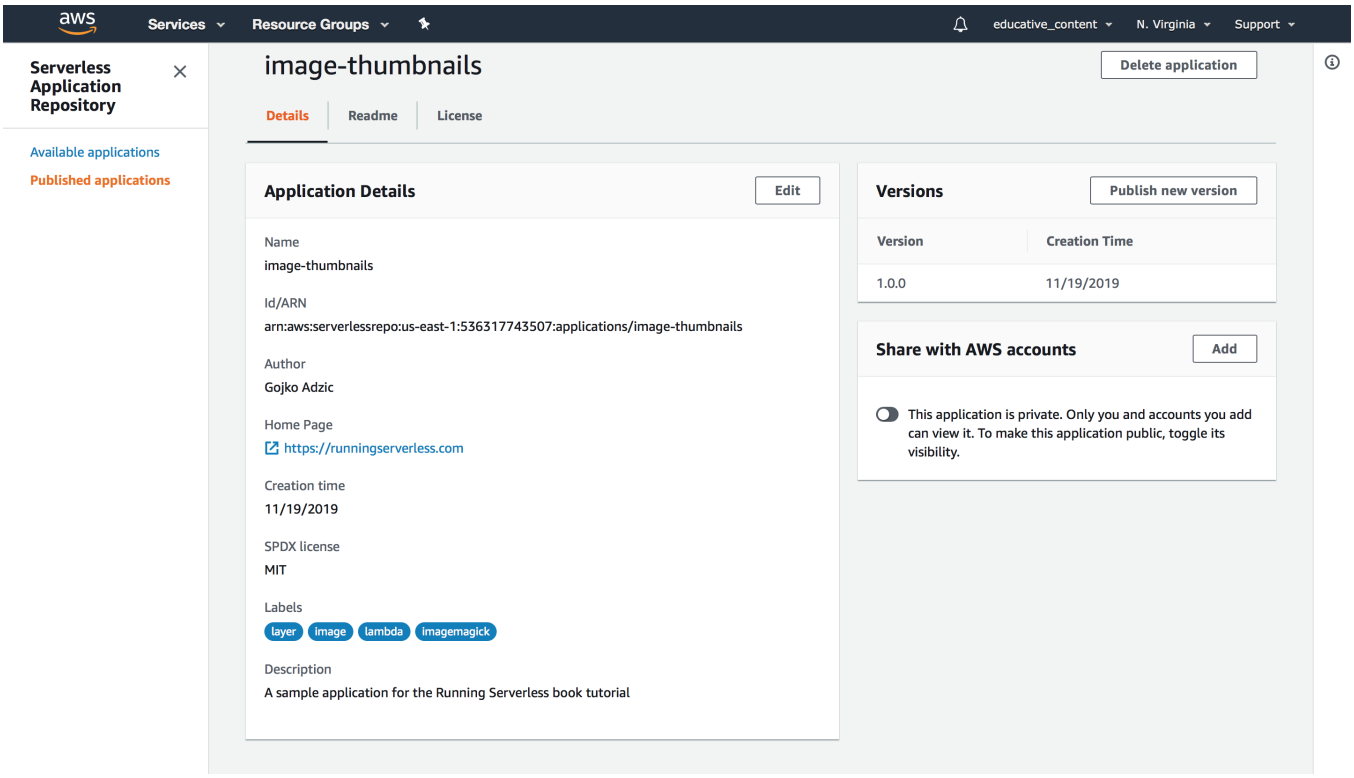
Key:	Value:
AWS_ACCESS_KEY_ID	Not Specified...
AWS_SECRET_ACCE...	Not Specified...
BUCKET_NAME	Not Specified...
AWS_REGION	Not Specified...

```
{
  "body": "{\"message\": \"hello world\"}",
  "resource": "/{proxy+}",
  "path": "/path/to/resource",
  "httpMethod": "POST",
```

```
  "isBase64Encoded": false,
  "queryStringParameters": {
    "foo": "bar"
  },
  "pathParameters": {
    "proxy": "/path/to/resource"
  },
  "stageVariables": {
    "baz": "qux"
  },
  "headers": {
    "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8",
    "Accept-Encoding": "gzip, deflate, sdch",
    "Accept-Language": "en-US,en;q=0.8",
    "Cache-Control": "max-age=0",
    "CloudFront-Forwarded-Proto": "https",
    "CloudFront-Is-Desktop-Viewer": "true",
    "CloudFront-Is-Mobile-Viewer": "false",
    "CloudFront-Is-SmartTV-Viewer": "false",
    "CloudFront-Is-Tablet-Viewer": "false",
    "CloudFront-Viewer-Country": "US",
    "Host": "1234567890.execute-api.us-east-1.amazonaws.com",
    "Upgrade-Insecure-Requests": "1",
    "User-Agent": "Custom User Agent String",
    "Via": "1.1 08f323deadbeefa7af34d5feb414ce27.cloudfront.net (CloudFront)",
    "X-Amz-Cf-Id": "cDehVQoZnx43VYQb9j2-nvCh-9z396Uhbp027Y2JvkCPNLmGJHqlaA==",
    "X-Forwarded-For": "127.0.0.1, 127.0.0.2",
    "X-Forwarded-Port": "443",
    "X-Forwarded-Proto": "https"
  },
  "requestContext": {
    "accountId": "123456789012",
    "resourceId": "123456",
    "stage": "prod",
    "requestId": "c6af9ac6-7b61-11e6-9a41-93e8deadbeef",
    "requestTime": "09/Apr/2015:12:34:56 +0000",
    "requestTimeEpoch": 1428582896000,
    "identity": {
      "cognitoIdentityPoolId": null,
      "accountId": null,
      "cognitoIdentityId": null,
      "caller": null,
      "accessKey": null,
      "sourceIp": "127.0.0.1",
      "cognitoAuthenticationType": null,
      "cognitoAuthenticationProvider": null,
      "userArn": null,
      "userAgent": "Custom User Agent String",
      "user": null
    },
    "path": "/prod/path/to/resource",
    "resourcePath": "/{proxy+}",
    "httpMethod": "POST",
    "apiId": "1234567890",
    "protocol": "HTTP/1.1"
  }
}
```


In a few moments you should see the publishing report, showing a link of where to check out the application in the repository. Open the link in the

browser, and you'll see the information about the published application.

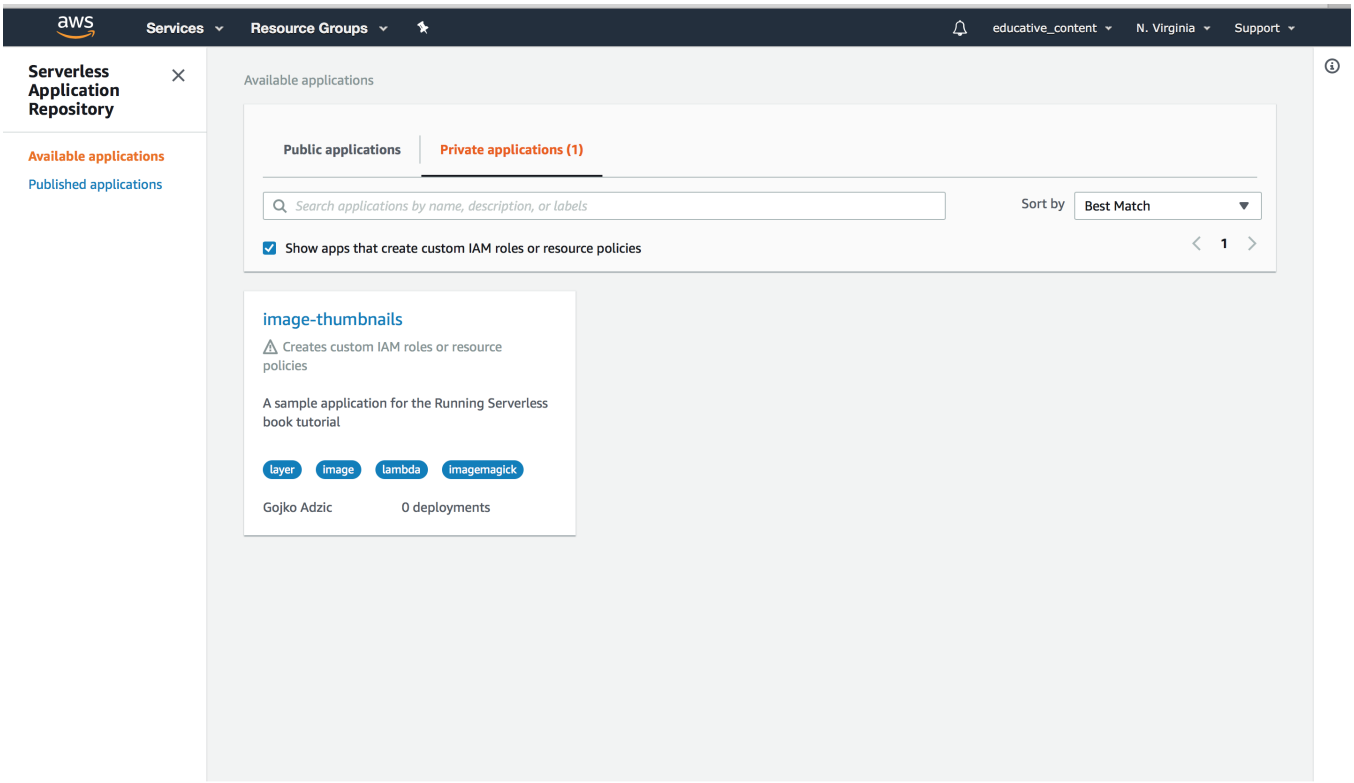


Our new application in the SAR.

Notice that the menu on the left of the figure above has two options:

- *Published applications* is the administrative part, allowing you to edit and republish an application. It's preferred to use command-line tools for that like you just did.
- *Available applications* is the client-side part, allowing you to browse and deploy existing applications.

Switch to the *Available applications* section then change to the *Private applications* tab (see figure below). Make sure you check the *Show apps that create custom IAM roles* button, and you'll see the application you just published.



The Private applications tab shows your personal applications in the SAR repository.

Click the title of the application to see the details, and scroll down to the bottom of the page. You will see the application options (see figure below). The repository automatically creates a configuration form for an application using the `Parameters` section of its SAM template, including custom validation rules and messages. At the very bottom, you'll also see a *Deploy* button, which you can use to create a new instance of the application in your account. Of course, deploying from a web page is a bit unnecessary if you have access to the source code of an application, but this is potentially useful for sharing components with external clients.

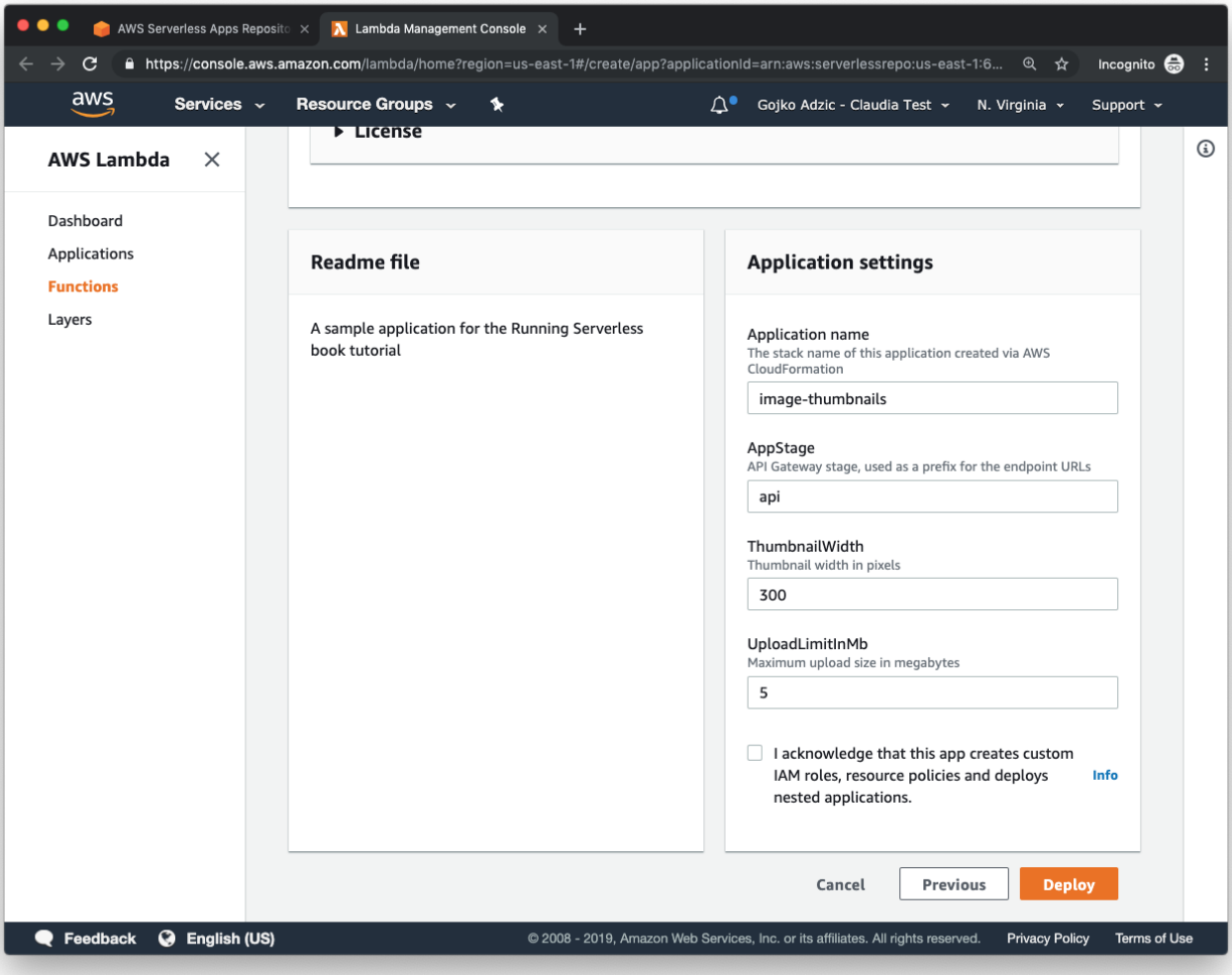
Using SAR or just sharing templates

CloudFormation templates produced by `sam package` fully describe a piece of infrastructure, and you can share them with other team members or even publish them on a website so other AWS users can install them. Templates published like that are locked to a particular region, and you'll need to use separate buckets and publish different templates for each available region. Public applications in the SAR are not restricted to a region, so users can easily choose where to deploy them.

After `sam publish`, the application is initially private, available only to your AWS account. You can share it with other accounts or even make it fully public, using the toggle button in the *Share with AWS accounts* section of the application administrative page. If you prefer to use the command line, share a published application by setting its access policy using the `aws serverlessrepo` command, such as the following one (replace `APP_ID` with your full application ARN):

```
aws serverlessrepo put-application-policy --application-id APP_ID --statem
ents Principals='*',Actions=Deploy
```

To share an application with only specific accounts instead of making it public, just list the account principals in the `Principals` part, instead of using an asterisk.



SAR created a configuration form (Application Settings on the right side of the page) based on the parameters from our application template.

That sums up this chapter. Now, you are ready to move on to the Interesting Experiments section in the next lesson.