Contents

List of Figures

List of Tables

1 Introduction

Screenshotbot is a screenshot testing service, designed for fast moving Mobile and Web development teams.

While Screenshotbot is a Software-as-a-Service (SaaS), we are also an open-source tool, and we also provide an Enterprise self-hosted service.

1.1 Audience for this document

This document is mainly geared toward DevOps or IT teams setting up Screenshotbot Enterprise on your own AWS Marketplace cloud. We'll touch on what's needed to integrate with your mobile teams, but the primary documentation for integration will be at `https://screenshotbot.io/documentation`.

This document will guide you on how to manage your Screenshotbot cluster: things like installing, upgrading and monitoring.

If you wish for us to manage your cluster, please send us an email at support@screenshotbot.io.

2 Architecture Description

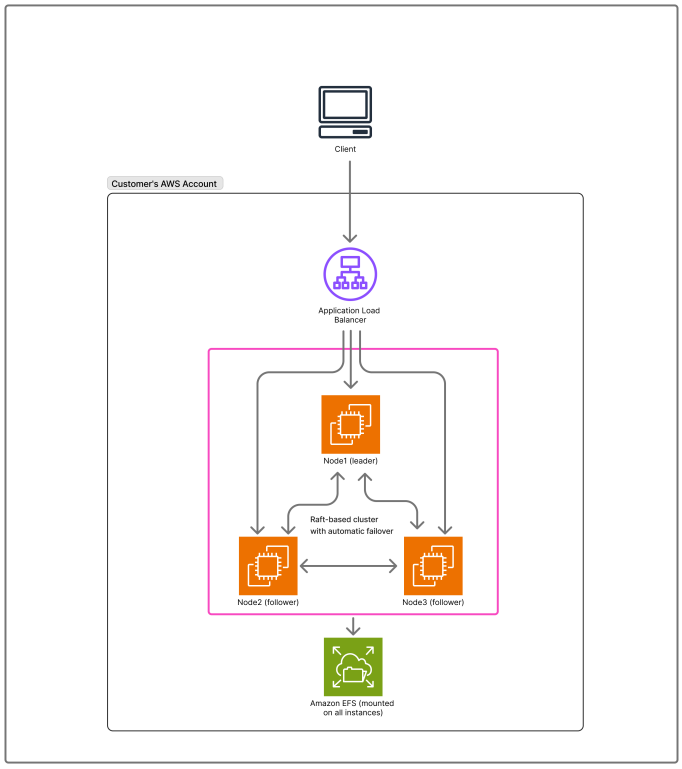

Figure 1: Overview of Cloudformation components

Screenshotbot runs primarily as a cluster of three (or five) servers. These servers replicate among each other using the Raft protocol. Each server has an identical set of information stored, but at any point of time only one of the servers will be a *leader* (the other servers will be called *followers* or *replicas*).

Screenshotbot is self-contained within this cluster: we do not use any external database. We do share an Elastic File System between the servers to store image data, and to store backups.

As long as a majority of the servers are alive (*quorum*), we will be able to respond to requests. If we lose quorum, the cluster gets into a read-only state.

The servers in the cluster automatically elect a leader, and any server in the cluster might be elected to be a leader at any time. You should not make assumptions about which server is the leader.

You may send write requests to any server, and it will forward it to the leader. However, to avoid a network-hop, our default configuration is designed to make

the Load Balancer always send the request directly to the leader. We will describe this later.

2.1 Datastore Snapshots

Screenshotbot's Raft cluster replicates transactions to each replica as soon as it happens. Once a day at 2AM of the server time, we also perform a *snapshot* of all the data on the server.

If any instance of the cluster is killed or rebooted, it will:

- First, load the snapshot back into memory
- then, replay the transaction logs until it is caught up

A leader in the snapshot state will hold write requests until the snapshot completes. The snapshot step can become significant in larger installations, but in most cases it will complete within 30 seconds.

2.2 Resource requirements

2.2.1 Memory

Screenshotbot is memory heavy: all of the data (except images, logs and some metadata) is stored in-memory. Because of the garbage collector, we typically don't advise consuming more than 50% of the system's available memory.

We recommending starting with a machine type with at least 4GB of RAM. As time goes by, and as you store more data in Screenshotbot you will need to increase the instance memory.

2.2.2 CPU

In general, Screenshotbot is not CPU heavy.

However, it needs spare CPU to handle image processing which might happen occassionally. (For instance, if a developer makes a UI change that affects thousands of screenshots, then we'll need to process each of them.) We recommend giving the instance some parallel compute capacity for image processing.

At the moment, Screenshotbot does not use the CPU of the followers for image processing.

Additional CPU cores can also help speed up the Snapshot process as described below.

2.3 Single-server configuration

For cost reasons, you may choose to run a Screenshotbot without any replicas (i.e. one server instead of three). This is an acceptable configuration for smaller companies, but needs additional considerations with respect to backups and downtime.

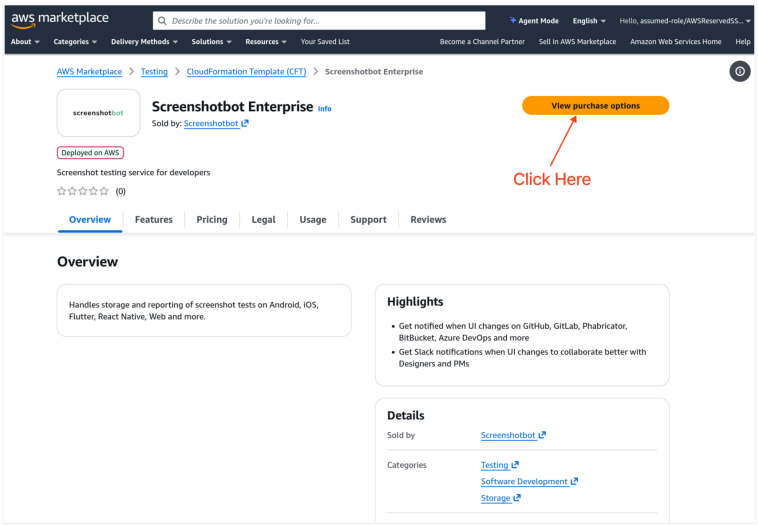

Figure 2: Click the link to view purchase options

3 Bring up a cluster via AWS Marketplace

Now that you understand the architecture, we're ready to initialize a cluster using AWS Marketplace.

This configuration creates a 3-instance cluster. If you wish to initialize a 1-instance cluster, follow this process and then follow the process to remove replicas from the cluster.

For a 5-instance cluster, follow this process and then use the process to add replicas to the cluster.

3.1 Run the CloudFormation

Screenshotbot is available on the AWS Marketplace at this link: `https://aws.amazon.com/marketplace/pp/prodview-w3zl3gheazmbk`.

At the time of writing this marketplace app is not public, and you need a specific invitation to access the application. If you are reading this, it is likely the app is already public.

First, click the "View purchase options" link on the Marketplace app to get started. (See Figure 2.)

Scroll to the bottom of this page and click "Subscribe" (See Figure 4). You will not be charged until you deploy your cluster.

Once you have subscribed to Screenshotbot Enterprise, you can Launch it via Cloudformation Stacks in your AWS Markedplace dashboard. (TODO: we'll add screenshots for these pages in a future version of this document.)

The Cloudformation stack does not create a dedicated VPC. If you wish to use

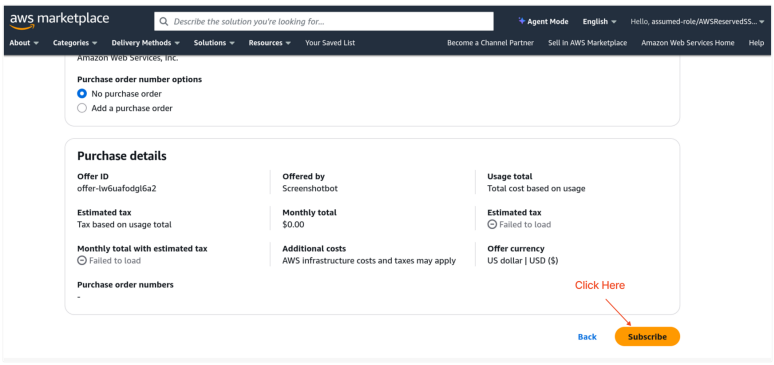

Figure 3: Click the Subscribe button

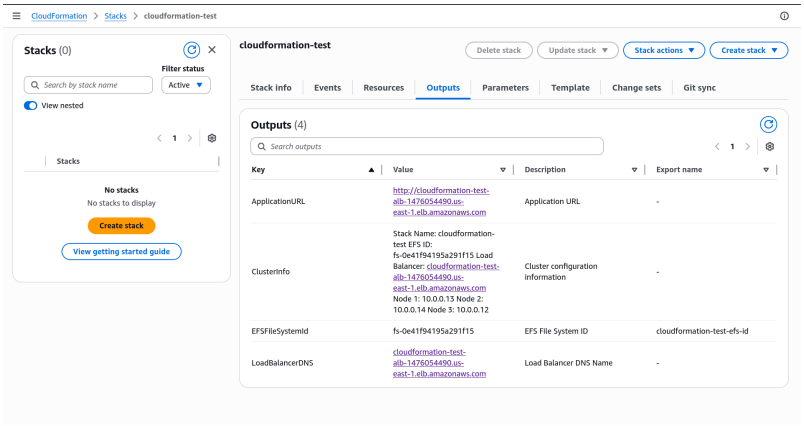

Figure 4: Outputs from the Cloudformation Stack

an isolated VPC, please create the VPC and two separate subnets for the load balancer.

By default, the stack also creates all the instances under the same subnet. We require a second subnet running in a different Availability Zone for the Load Balancer.

As we'll see later, you'll have the ability to create a cluster with each instance in a different AZ if you so choose.

Once you stack is created, take a look at the Outputs of the stack (Figure **??**). Clicking the Application URL will take you to the Login page for Screenshotbot (Figure 5).

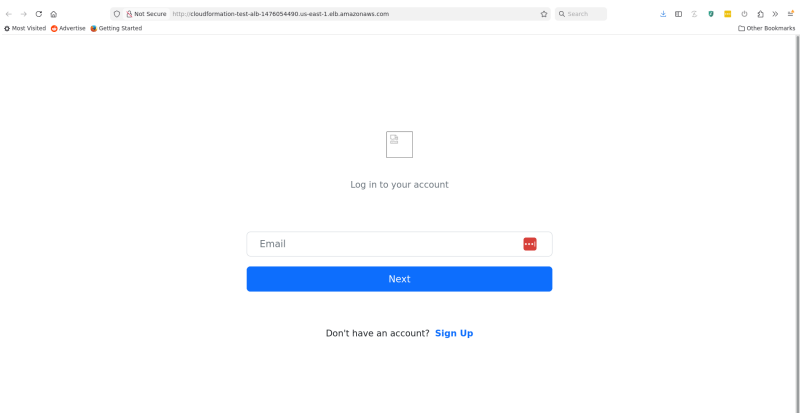

Figure 5: The default login page

4 CLI administration

Before we can continue, SSH to one of the machines in the cluster as `admin` user. For example, `ssh admin@x.y.z.w`.

Once you're in the machine, we provide a CLI wrapper called `screenshotbot`. This will be the tool you can use to configure the cluster.

```
admin@node1~$ screenshotbot cluster status
Server PID: 605
Leader: 192.168.57.10
Is current node Leader?: T
Peers:
  192.168.57.10
  192.168.57.11
  192.168.57.12
```

Depending on whether the machine you SSH-ed to was a leader or a follower, you'll get a different output. For instance, here's the output from a follower:

```
admin@node2:~$ screenshotbot cluster status
Server PID: 607
Leader: 192.168.57.10
Is current node Leader?: NIL
Peers:
```

4.1 Switching Leaders

Many of the commands listed in this document needs to be run on the leader. One way to do this is to SSH to an arbitrary machine, run `screenshotbot cluster status` to find the leader and then SSH to that leader.

Alternatively, once you SSH to a machine, just switch that machine to be the leader:

```
admin@node2:~$ screenshotbot cluster leadership acquire
Requesting leadership for this node (PID 607)...
Vote initiated.
Got result: NIL, NIL
admin@node2:~$ screenshotbot cluster status
Server PID: 607
Leader: 192.168.57.11
Is current node Leader?: T
Peers:
  192.168.57.10
  192.168.57.11
  192.168.57.12
Got result: NIL, NIL
```

5 Setting up a custom domain name and HTTPS

At this point, we can only access the Screenshotbot server via HTTP and via the default domain name provide by the Application Load Balancer.

To set up a custom domain, first set up a `CNAME` to the ALB domain name from your DNS manager, let's say this domain is `screenshotbot.example.com`. (The details of how to set this up are out of scope of this document, but ping our support if you need help.)

Next create a certificate for the domain in AWS Certificate Manager. Follow the instructions in ACM to validate the certificate.

The Cloudformation scripts have created a single listener in the Load Balancer. Find this load balancer and change the protocol to HTTPS with port 443.

At this point, you should be able to access `https://screenshotbot.example.com` from your web browser.

5.1 Updating the default domain in Screenshotbot

Screenshotbot needs to know the default domain name in order to generate URLs for emails and external notifications such as Slack.

To do this, ssh to the leader of the cluster and run the following command

```
admin@node1$ screenshotbot config set \
     --key installation.domain \
     --value https://screenshotbot.example.com
```

5.2 [Optional] Redirect HTTP to HTTPS

We recommend creating a new HTTP listener that just redirects all requests to HTTPS.

6 Setting up Single Sign-On (SSO)

Screenshotbot supports Single Sign-On with OpenID Connect (OIDC). Most Identity Providers (including Okta and Microsoft Entra) support OIDC.

If you choose to use SAML, we suggest using an intermediate Amazon Incognito instance. In this case Screenshotbot will authenticate against the Incognito, which in turn will authenticate via SAML to your Identity Provider.

6.1 Creating an OIDC App

In your Identity Provider (such as Okta, Microsoft Entra, Google Workplace), create a new OpenID Connect Application.

The OIDC Callback should be `https://screenshotbot.example.com/account/oauth-callback`.

The scope needs to be "openid,email,profile". (Each provider configures the scopes differently.

If you're looking for a logo, you can use these images: `https://cdn.screenshotbot.io/assets/images/logo-dark.svg` or `https://cdn.screenshotbot.io/assets/images/logos/text-logo-with-vertical-space.png` or `https://cdn.screenshotbot.io/assets/images/logo-favicon.png` depending on context.

Microsoft Entra

For Microsoft Entra, you don't explicitly add the `profile` scope. Instead, just add the appropriate claims: `email`, `first_name` and `family_name`.
In addition, we suggest adding the *User.Read* API permission: `https://learn.microsoft.com/en-us/entra/identity-platform/quickstart-configure-app-access-web-apis`. This permission is required to fetch profile pictures.

6.2 Configuring SSO parameters in Screenshotbot

To set up SSO, you need to set up the following three config values

- `sso.oidc.issuer`
- `sso.oidc.client-id`
- `sso.oidc.client-secret`

For example, you can set up the parameters like so:

```
admin@node1$ screenshotbot config set \
     --key sso.oidc.issuer \
     --value https://foobar.okta.com/
```

Once you set up the three parameters, you should no longer see the login screen when you go to `https://screenshotbot.example.com`, instead it should redirect you to the login screen for your Identity Provider.

7 Modifying the cluster

There are many reasons you might want to modify the cluster. For instance, you might want to migrate the cluster to a different availability zone, or you might need to upgrade the cluster to an updated AMI.

7.1 Adding replicas

To add a replica, create a new instance in the same security group with the appropriate AMI. Then, run:

```
admin@node1:~$ screenshotbot cluster add-peer 10.0.0.2
```

At this point the replica will start loading the snapshot from the leader. For larger installations this might take a while. You can track the progress of the replication by SSH-ing to the new replica and running:

```
admin@node4:~$ sudo journalctl -u screenshotbot -f
```

7.2 Removing replicas

Similarly to adding a replica, you can remove a replica with the following command on the leader:

```
admin@node1:~$ screenshotbot cluster remove-peer 10.0.0.2
```

7.3 Switching Leaders

As described previously, you can switch leaders by SSH to the replica you wish to be the leader. Then you can run the following command:

```
admin@node2:~$ screenshotbot cluster leadership acquire
```

8 Upgrading the cluster

Upgrading the cluster goes through the following steps. Then, connect to the current leader of the cluster and run:

- Once you have the new AMI id, add three new replicas with the new AMI
- Switch the leader to the new replicas
- Removed the old replicas

We recommend reading through the Release Notes for the release to make sure you follow any additional steps that might be required for the given release.