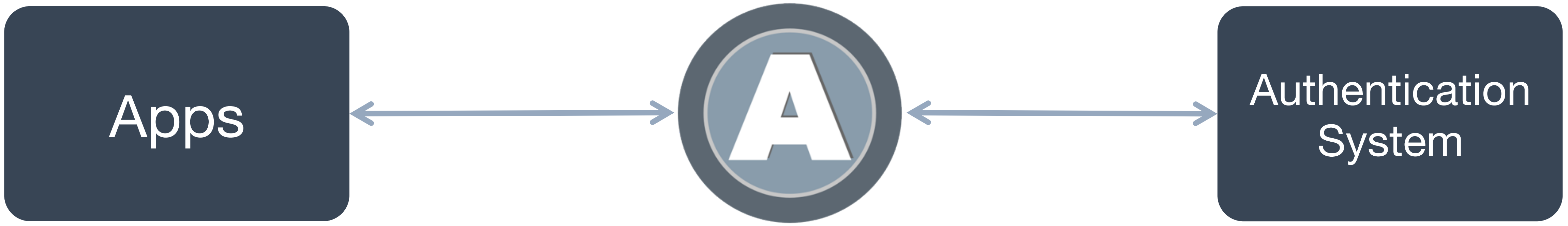
Building a Server Appliance with Node.js

Auth0

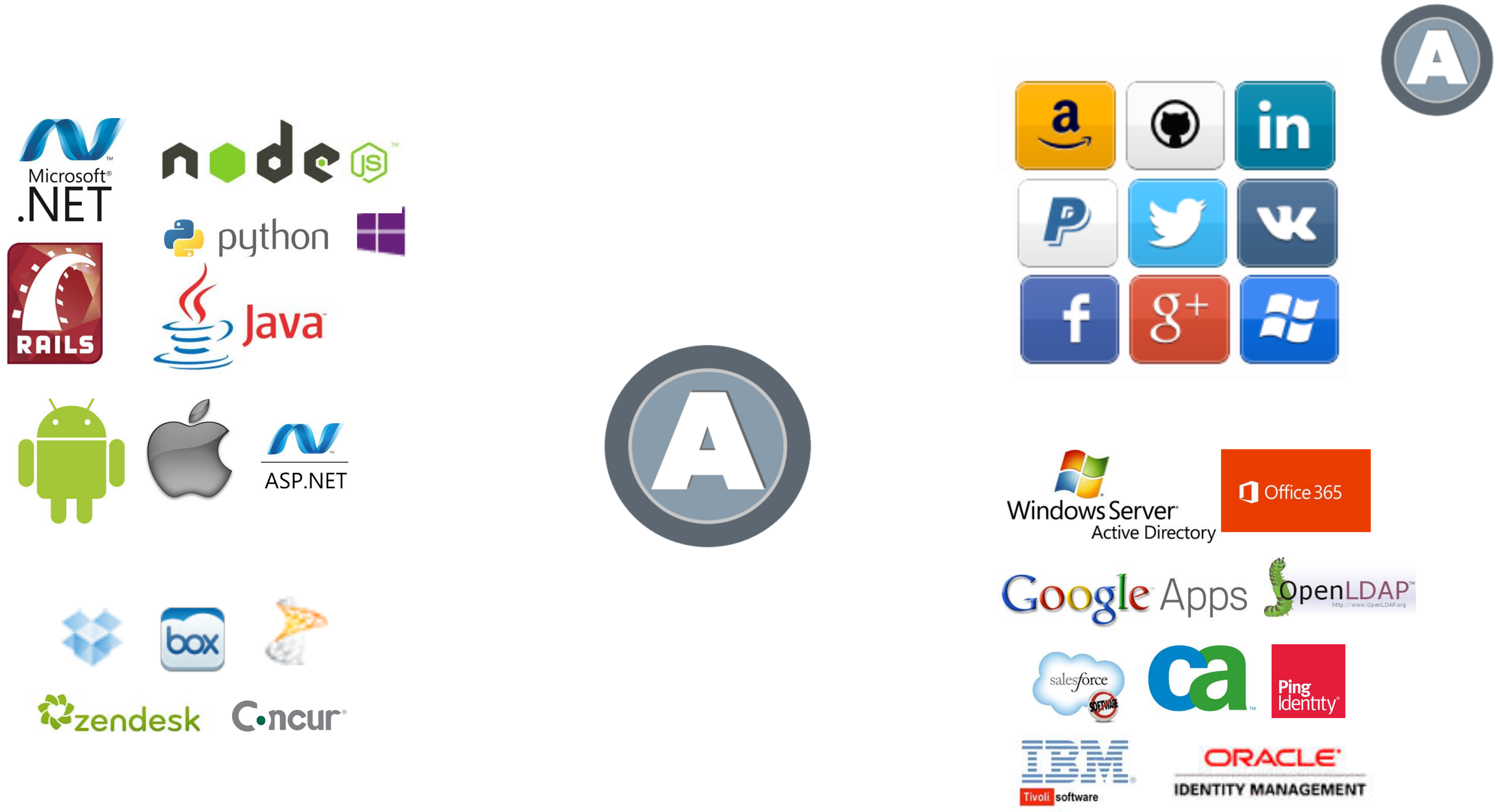
Eugenio Pace – eugeniop@auth0.com

@authzero @eugenio_pace


Apps
Authentication System

OAuth 1, 2
OpenId Connect
SAML 2.0
WS-Federation
WS-Trust

SAML
JWT
SWT


Microsoft .NET
node JS
python
RAILS
Java
ASP.NET
box
zendesk
Concur
Windows Server Active Directory
Office 365
Google Apps
OpenLDAP
salesforce
ca
Ping Identity
IBM Tivoli software
ORACLE IDENTITY MANAGEMENT

SSO

Directory

Log
Audit

SSO

Directory

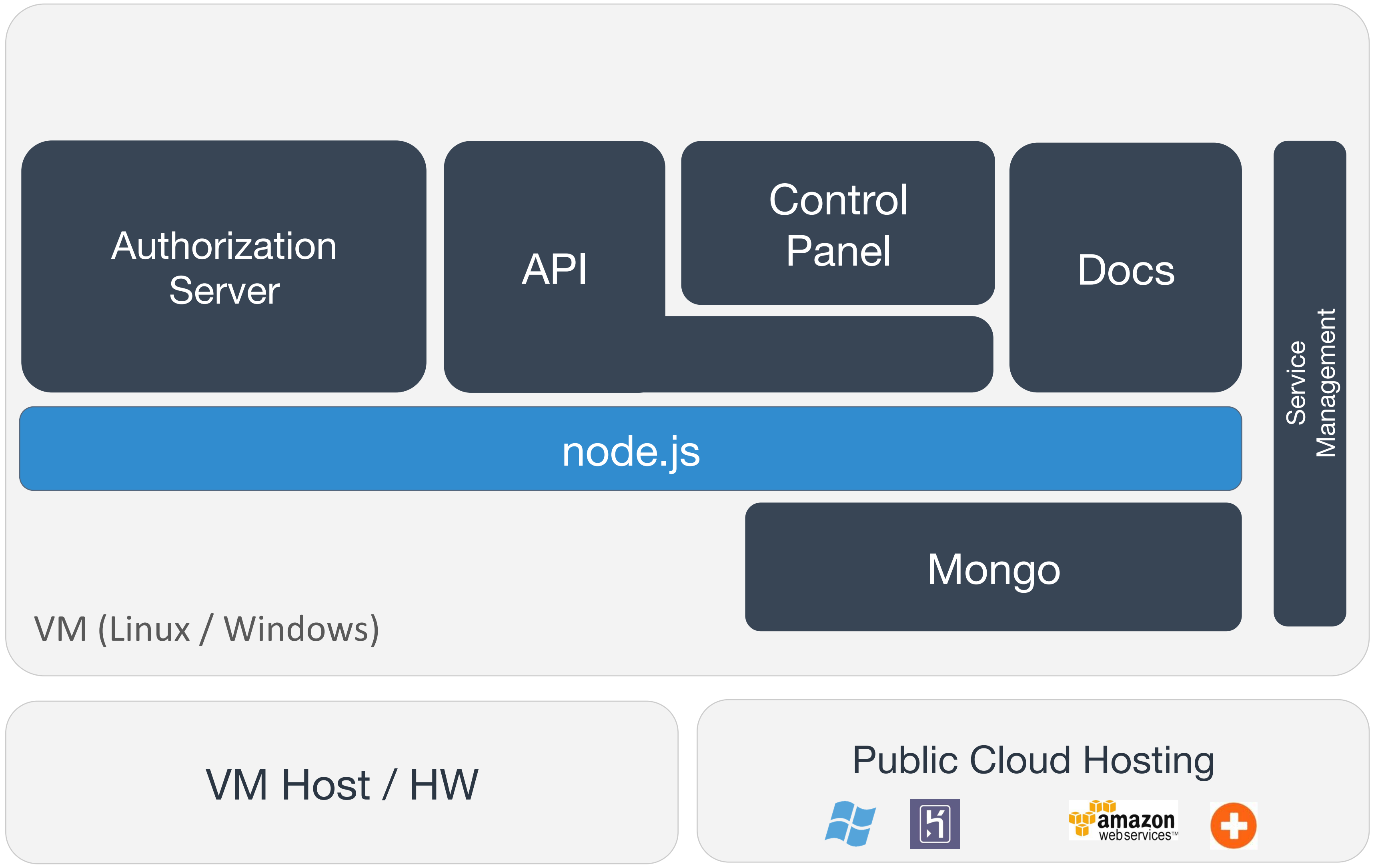
Log
Audit

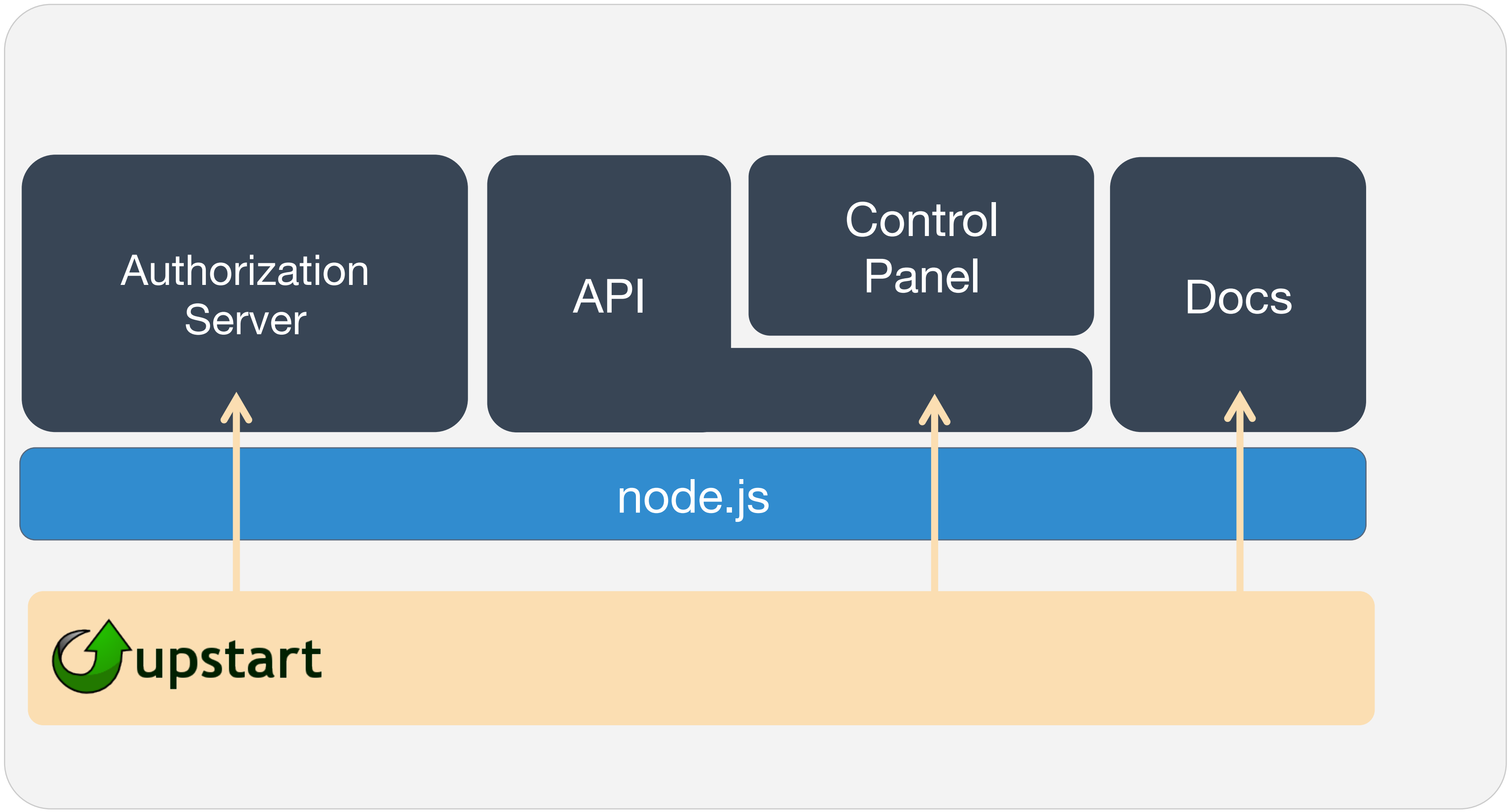
Setup

SDK

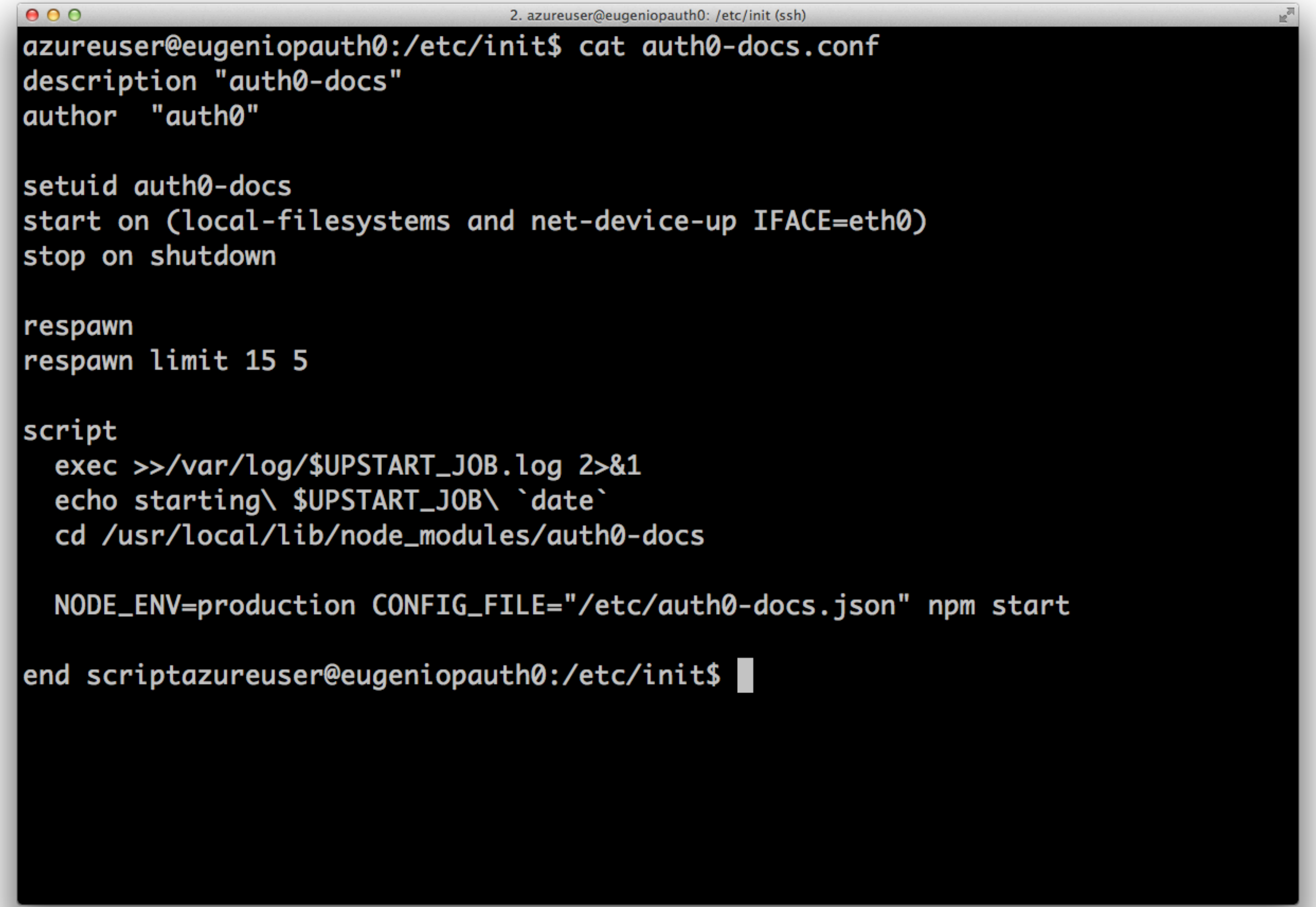
Run
Anywhere

Demo!


Authorization Server
API
Control Panel
Docs
Service Management
node.js
Mongo
VM (Linux / Windows)
VM Host / HW
Public Cloud Hosting
amazon webservices



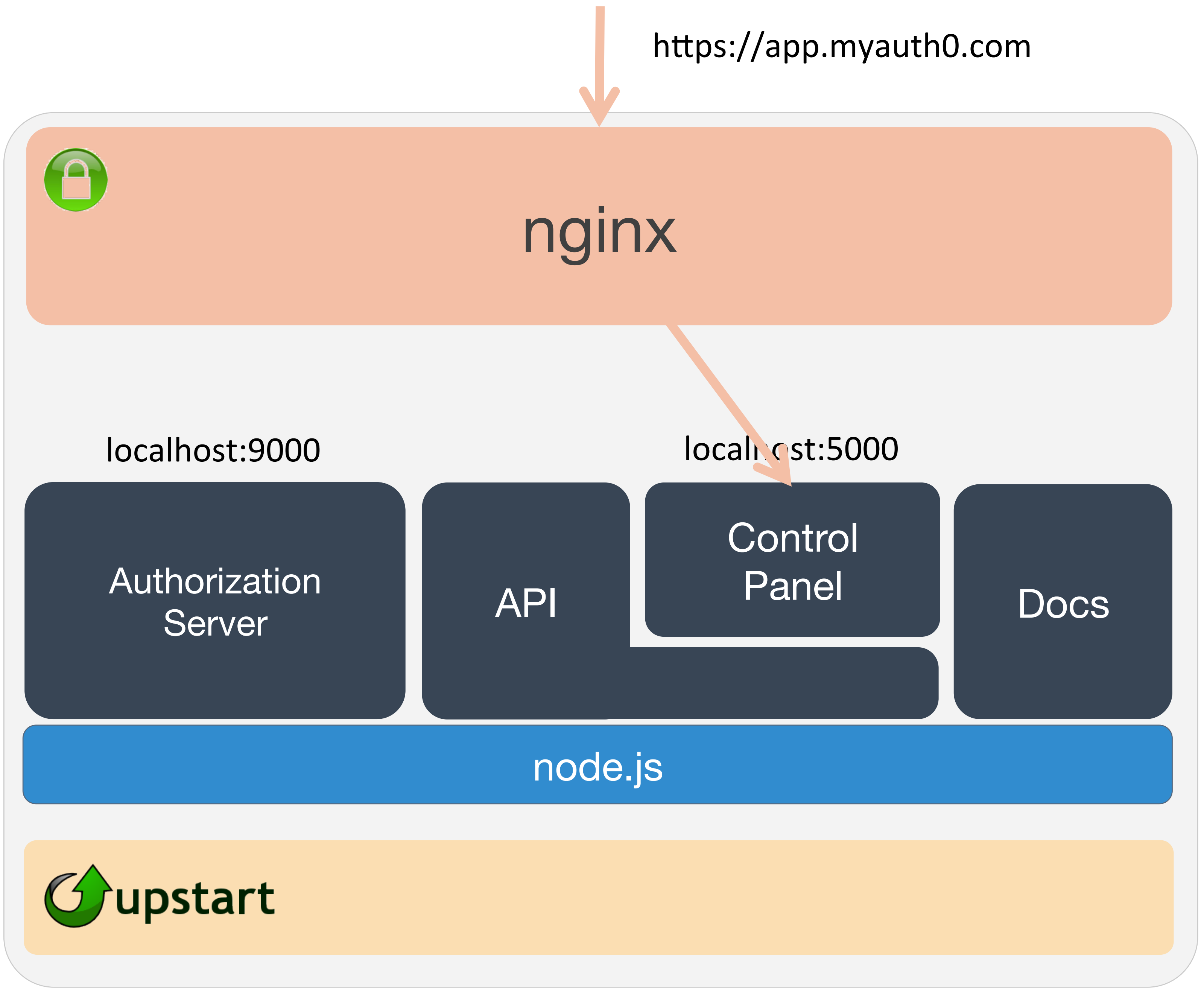
http://upstart.ubuntu.com/

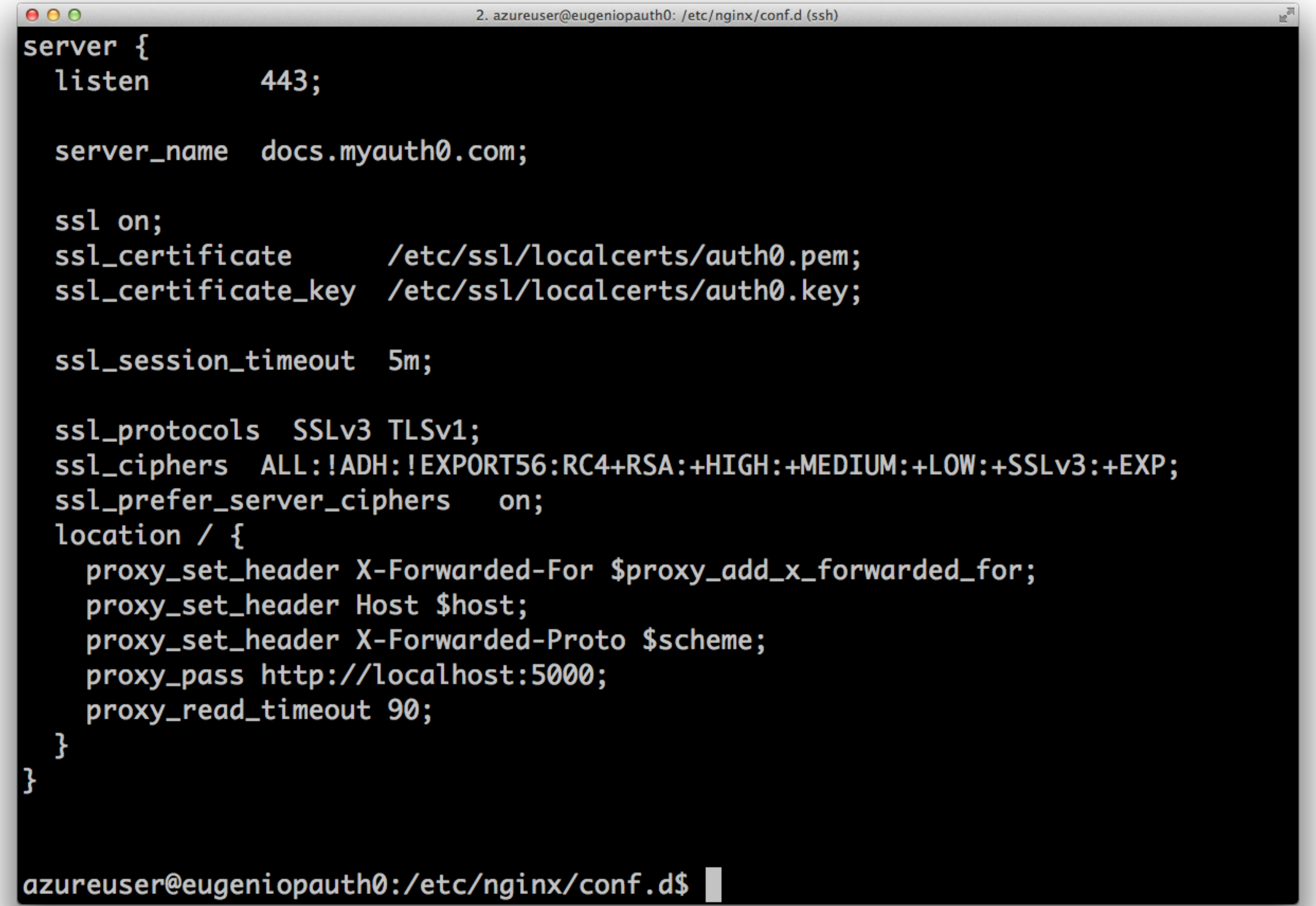


```
azureuser@eugeniopauth0:/etc/init$ cat auth0-docs.conf
description "auth0-docs"
author  "auth0"

setuid auth0-docs
start on (local-filesystems and net-device-up IFACE=eth0)
stop on shutdown

respawn
respawn limit 15 5

script
  exec >>/var/log/$UPSTART_JOB.log 2>&1
  echo starting\ $UPSTART_JOB\ `date`
  cd /usr/local/lib/node_modules/auth0-docs

  NODE_ENV=production CONFIG_FILE="/etc/auth0-docs.json" npm start

end scriptazureuser@eugeniopauth0:/etc/init$ 
```

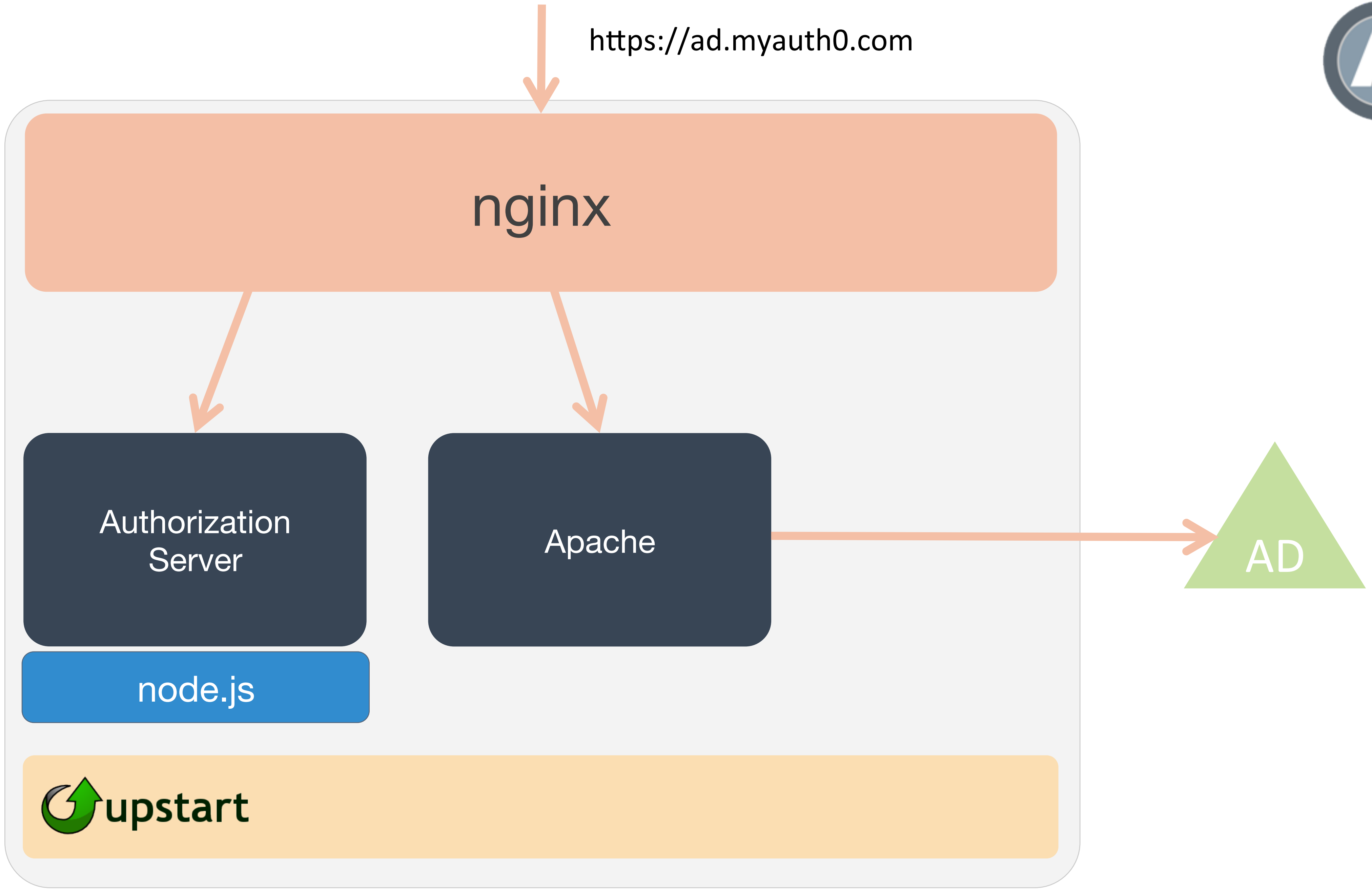
Considering…

- forever → https://github.com/nodejitsu/forever


https://app.myauth0.com
nginx
localhost:9000
localhost:5000
Authorization Server
API
Control Panel
Docs
node.js
upstart



```
server {
  listen       443;

  server_name  docs.myauth0.com;

  ssl on;
  ssl_certificate      /etc/ssl/localcerts/auth0.pem;
  ssl_certificate_key  /etc/ssl/localcerts/auth0.key;

  ssl_session_timeout  5m;

  ssl_protocols  SSLv3 TLSv1;
  ssl_ciphers  ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv3:+EXP;
  ssl_prefer_server_ciphers   on;
  location / {
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $host;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_pass http://localhost:5000;
    proxy_read_timeout 90;
  }
}


azureuser@eugeniopauth0:/etc/nginx/conf.d$
```


https://ad.myauth0.com
nginx
Authorization Server
Apache
node.js
upstart
AD

Considering…

- node-http-proxy → https://github.com/nodejitsu/node-http-proxy
- haproxy

Getting the bits

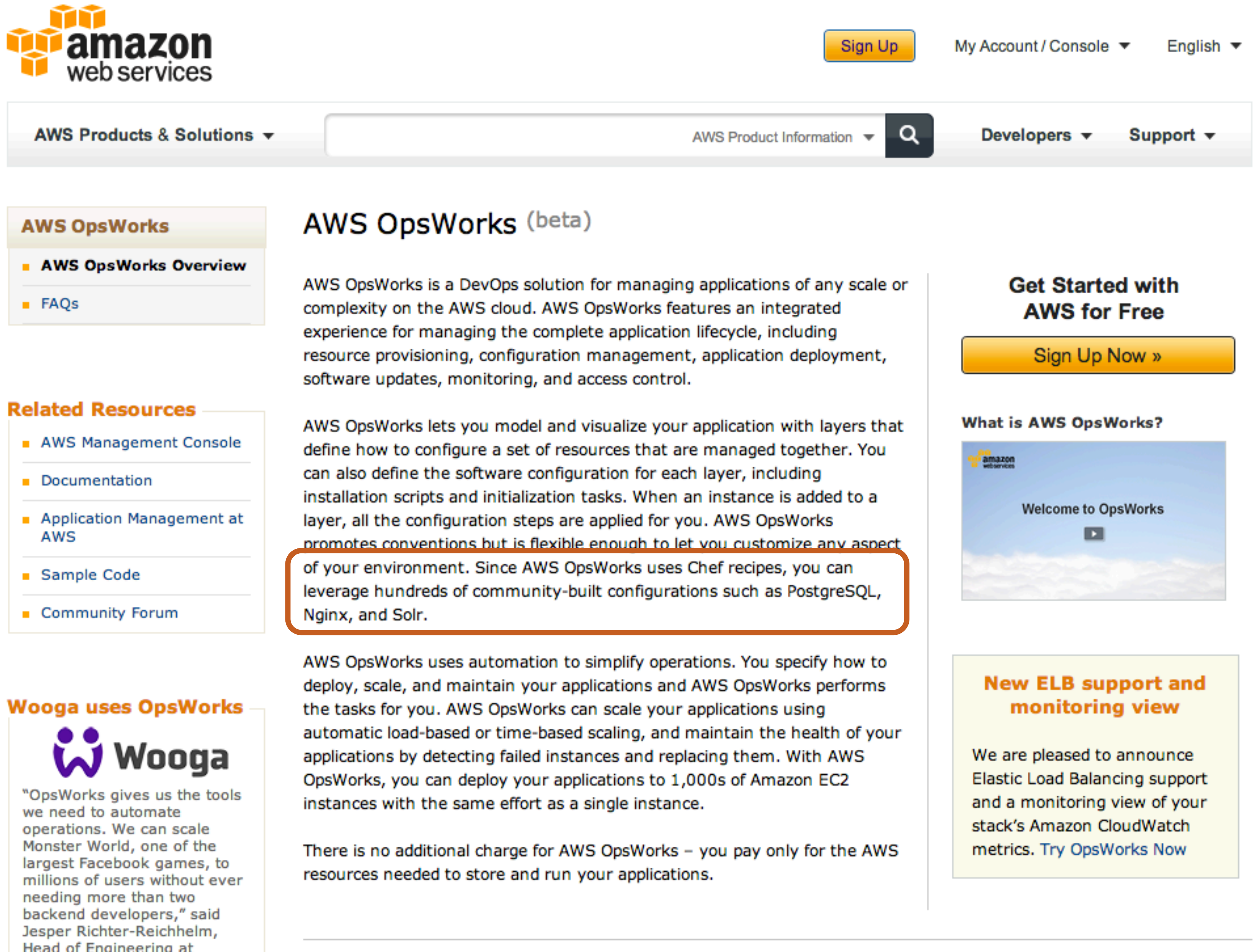
1. `sudo apt-get install puppetmaster`
2. `wget https://builds.auth0.com/{build}`
3. `sudo puppet apply --modulepath=modules -e "include run”`
4. MyAuth0!

1. Generally well documented
2. Good support
   https://puppetlabs.com/

3. Minimize content in you package
4. Leverage install tools (apt, npm)

Considering…

- chef → http://www.opscode.com/chef/



AWS OpsWorks

- AWS OpsWorks Overview
- FAQs

Related Resources

- AWS Management Console
- Documentation
- Application Management at AWS
- Sample Code
- Community Forum

Wooga uses OpsWorks

Wooga

"OpsWorks gives us the tools we need to automate operations. We can scale Monster World, one of the largest Facebook games, to millions of users without ever needing more than two backend developers," said Jesper Richter-Reichhelm, Head of Engineering at

AWS OpsWorks (beta)

AWS OpsWorks is a DevOps solution for managing applications of any scale or complexity on the AWS cloud. AWS OpsWorks features an integrated experience for managing the complete application lifecycle, including resource provisioning, configuration management, application deployment, software updates, monitoring, and access control.

AWS OpsWorks lets you model and visualize your application with layers that define how to configure a set of resources that are managed together. You can also define the software configuration for each layer, including installation scripts and initialization tasks. When an instance is added to a layer, all the configuration steps are applied for you. AWS OpsWorks promotes conventions but is flexible enough to let you customize any aspect of your environment. Since AWS OpsWorks uses Chef recipes, you can leverage hundreds of community-built configurations such as PostgreSQL, Nginx, and Solr.

AWS OpsWorks uses automation to simplify operations. You specify how to deploy, scale, and maintain your applications and AWS OpsWorks performs the tasks for you. AWS OpsWorks can scale your applications using automatic load-based or time-based scaling, and maintain the health of your applications by detecting failed instances and replacing them. With AWS OpsWorks, you can deploy your applications to 1,000s of Amazon EC2 instances with the same effort as a single instance.

There is no additional charge for AWS OpsWorks – you pay only for the AWS resources needed to store and run your applications.

Get Started with AWS for Free

Sign Up Now »

What is AWS OpsWorks?

New ELB support and monitoring view

We are pleased to announce Elastic Load Balancing support and a monitoring view of your stack's Amazon CloudWatch metrics. Try OpsWorks Now

On node

- Transactions
- Malleability
- No wheel reinventing

- *Service and Server*
- *Any cloud*

Thanks!

eugeniop@auth0.com