# Delivering and Deploying Turn-Key Self-Orchestrating Solutions

Docker Seattle Meetup
August 17, 2016

Bob Dickinson

Synchro Labs, Inc.

https://synchro.io

# Synchro Labs, Inc.

https://synchro.io SynchroLabs @synchrolabs Redmond, WA

## Founding Team

Bob Dickinson
bob@synchro.io
@rdd3
BobDickinson

Blake Ramsdell
blake@synchro.io
@blake182
bcr

# Alternate Profile Pic

with Ranger, my pet llama

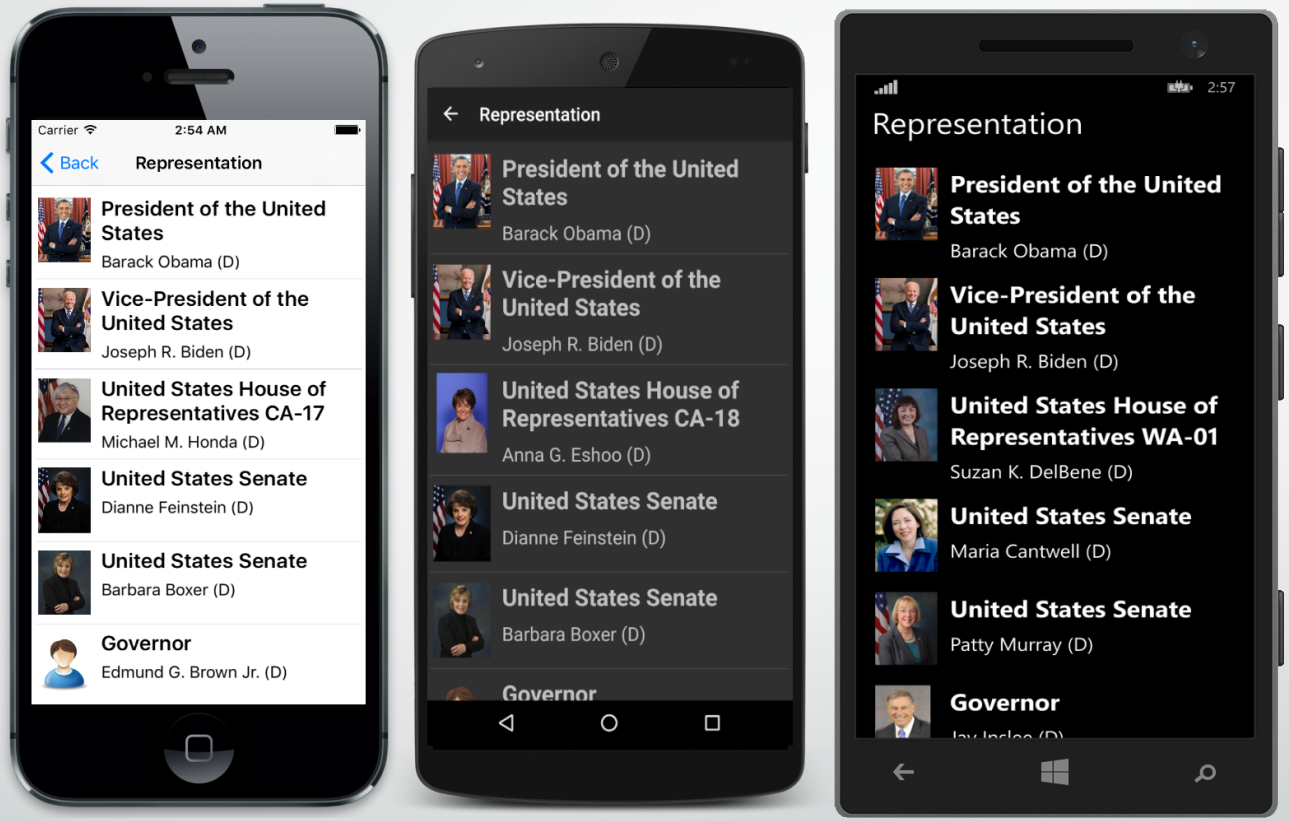
# What is Synchro?

- Mobile app development platform based on **JavaScript** and **Node.js**
- **Native apps** for Android, iOS, Windows, and Windows Phone, and web apps (mobile-optimized and desktop) - all from a single code base
- **Enterprise focus** (integration, dev ops, security)
- **Open source** server and web client, free native mobile dev client
- Custom native mobile apps **licensed for enterprise** deployment

# How does it Work?

- All of your mobile app code, including UX definition and interaction logic, runs on Synchro server under Node.js
- We remote the UX so that it runs on the mobile device and presents as a native mobile client
- Apps present as native to end users, but as Node/web apps to developers and DevOps

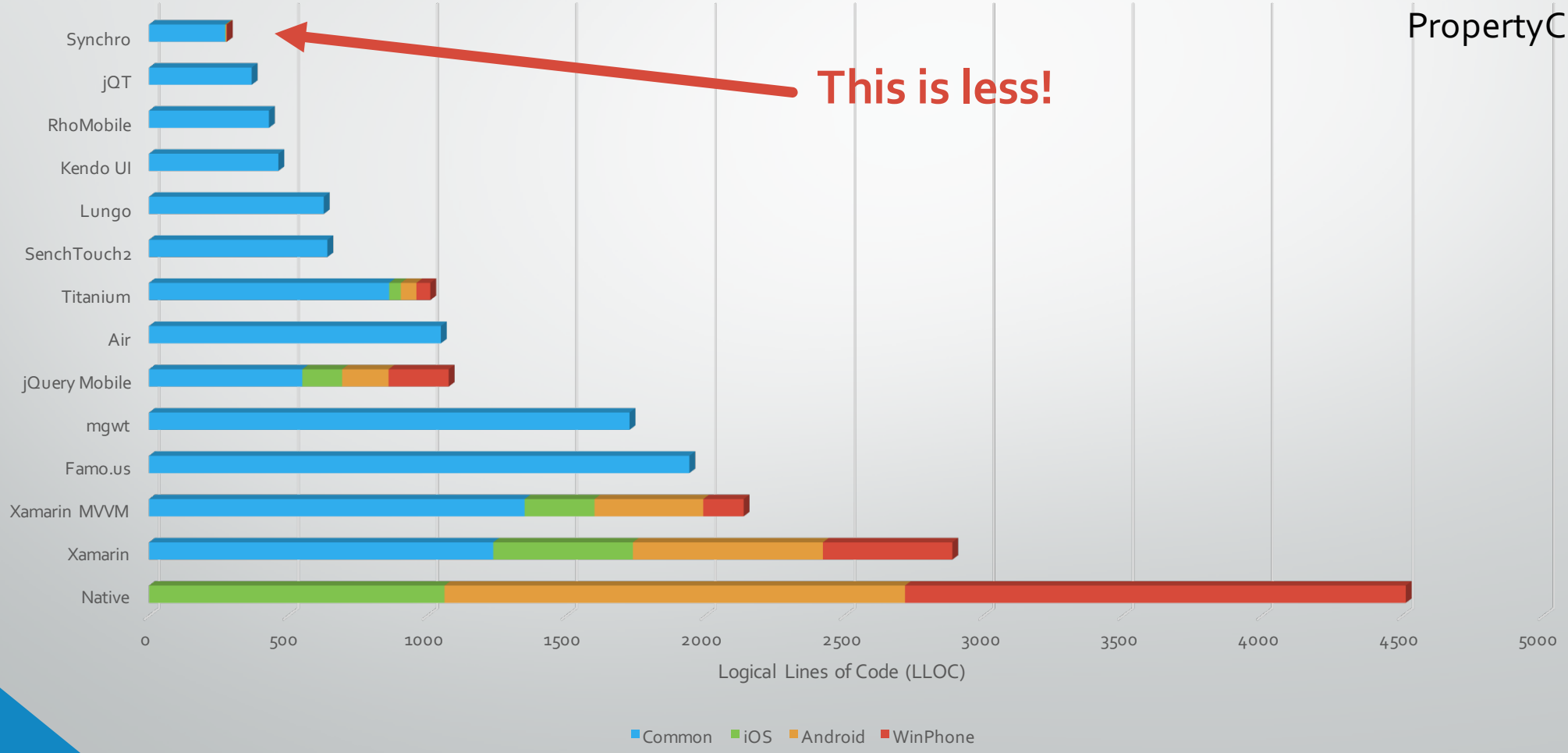
# Great Looking Native Apps




Synchro Civics shown above – get it in any App Store

200~ lines of code, all mobile platforms + web, running from the cloud

# Much Less Code




PropertyCross App

# Everything Runs in the Cloud

- Radically simplifies application design
  - Mobile app code runs on server - no more client/server!
  - App code can access enterprise resources directly, auth as app/server
- Provably secure
  - No app code ever runs on the mobile client
  - All communication between mobile client and server is always secure
  - Mobile client never communicates with anything other than server
- Ability to control access, instantly deploy updates, etc.

# For More Information…

- If you have enterprise apps that you'd like to mobilize
- Especially if you already know and love Node.js
- Take a look at Synchro: https://synchro.io
  - Getting started guide, tutorial, samples, documentation, and videos.

# Synchro Container and Orchestration Support

# Step 1: The Container

(AKA: “Yay, a Dockerfile!”)

```
FROM node:argon

# Create app directory
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app

# Bundle app source
COPY . /usr/src/app

# Install deps
RUN npm install
RUN cd synchro-apps && npm install

ENV SYNCHRO__PORT 80
EXPOSE $SYNCHRO__PORT

CMD [ "node", "app.js" ]
```

# Step 2: Orchestration Support

- Provide turn-key self-orchestrating solutions to our users/customers
- Migrate our own production services from managed web app (on Azure) to orchestrated containers/microservices (on Joyent)

# The AutoPilot Pattern using ContainerPilot

https://www.joyent.com/containerpilot

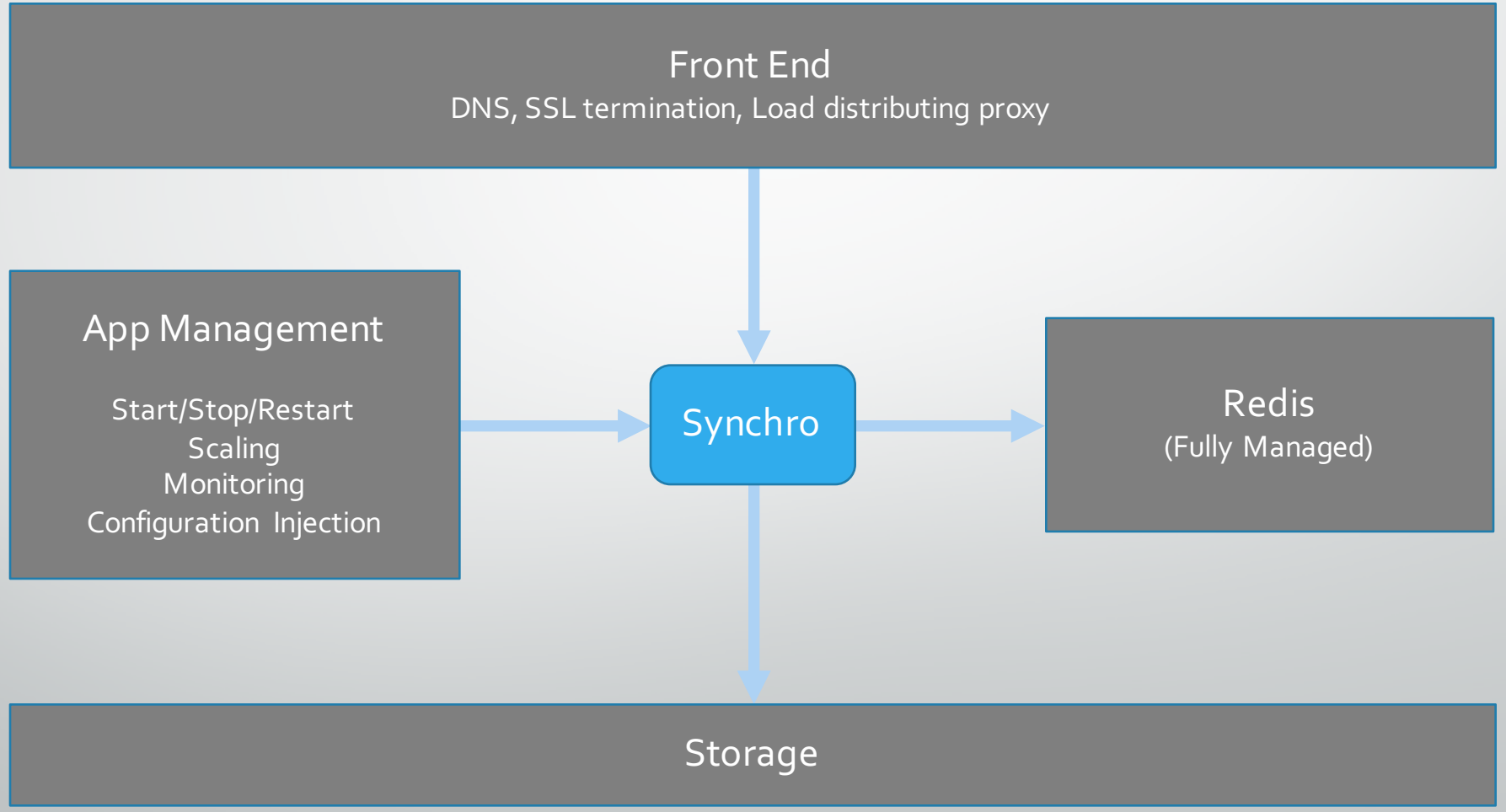
**"App-centric micro-orchestration"**

- Containers collaborate to adjust to each other automatically (they run on "AutoPilot")
  - Works with any orchestration solution **<< MUST HAVE**
  - Very easy to test/develop with production environment **<< BONUS**

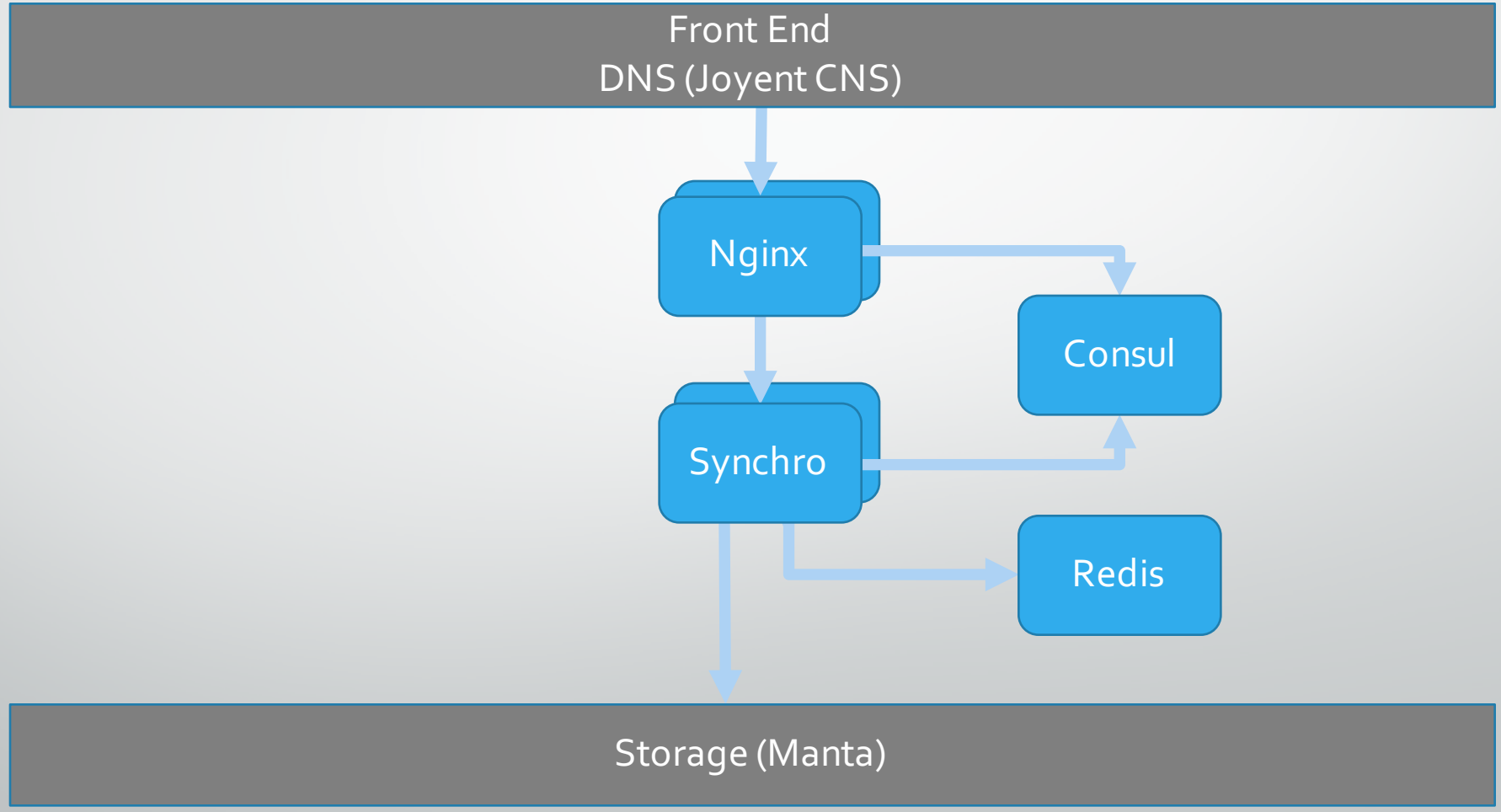
# A Case Study in Migrating from Managed Services to Orchestrated Micro-Services:

# Synchro Labs' Production API Server

# Synchro in a Managed Services Environment (Azure)



# Synchro in a Micro-Services Environment (Joyent)



# Nginx: The Beast that Must be Tamed

- Load-distributing proxy
  - Session affinity
- SSL termination (optional, configurable)
  - Tuned for security, performance (app-specific)
  - 301 Redirect of http to https
- Static file caching (migrated from CDN in managed environment)
- Websocket support
- **Goal:** Ability to replace entire nginx.conf (template) if needed via config

# Basic App Requirements

- Health Check
  - Endpoint should be configurable, consider allowing localhost access only
- Signal Handling
  - SIGINT for shutdown
  - SIGHUP if your app reconfigures (especially based on services from consul)
- Flexible Configuration Source
  - From environment vars or file
  - If from file, location of file should be configurable

# Configuration Injection

- Runtime
  - Environment variables
  - Volumes
- Build-time
  - Environment variables
  - Files

- We want our solution to be non-opinionated about how configuration is injected
- We want to be able to use public containers, and also support custom/private containers with baked-in state
- Some of our apps/containers themselves aren't very flexible about how they get configured (I'm looking at you nginx)
- Binary files, or even multiline text files (like cert/key), can be tricky to inject via environment vars

# Flexible Configuration Injection

- The ContainerPilot **preStart** hook is a great place to process config
- Recommendation: For any file-based config…
  - The local file path should be configurable (via an environment var)
  - There should be a way to inject file contents as a base64 encoded environment var

```
#!/bin/sh
preStart()
{
  : ${SSL_CERTS_PATH:="/etc/ssl/certs/ssl.crt"}
  if [ -n "$SSL_CERTS_BASE64" ]; then
    echo $SSL_CERTS_BASE64 | base64 -d > $SSL_CERTS_PATH
  fi;

  # >>> Process ssl.key, any other files as above

  # Rest of preStart logic (consul-template, etc)
}
```

# Configuration Packaging/Consolidation

- Package all configuration in one location (in our case: cloud storage)
- We need access to files – why not just use volumes?
  - Support for volume drivers on cloud storage is spotty/early (platform dependent)
    - No volume driver for Manta storage (yet)
  - Significant limitations on how volumes can be used in production
  - Can be difficult to access actual production data from dev using volumes
  - **Goal:** Non-opinionated solution to get files from anywhere, at runtime, with no client tooling (custom volume driver, etc) and no micro-service tooling (cli tools, etc) required

# StashBox

https://github.com/SynchroLabs/StashBox

- An http server/proxy that serves files from configurable endpoints
  - Supports environment vars, local files, http proxy, and cloud storage: Amazon (S3), Azure, Google, HP, IBM BlueMix, Joyent (Manta), OpenStack, and RackSpace
- Can be configured via environment (runtime) or config file (build-time)
- Clients require no tooling (just curl)
- Available from Docker Hub: synchro/stashbox
- Works great with AutoPilot/ContainerPilot

# Configuring StashBox Mounts

```
# StashBox config.json
{
  "mounts":
  [
    {
      "mount": "/",
      "provider": "file",
      "basePath": "stash"
    },
    {
      "mount": "ssl/domain.key",
      "provider": "env",
      "var": ”SSL_KEY_B64",
      "encoding": "base64"
    }
  ]
}
```

# Configuration Consolidated via StashBox

- Apps load their config files from StashBox in ContainerPilot preStart script
  - No tooling required – using curl
- The only configuration we have to inject is to point Nginx and Synchro to StashBox for their config files, and to point StashBox to Manta
- ALL application state is now in one place – on Manta
  - Easy to manage/control, very portable
  - Production test, staging, migration all simplified

# Our Environment (runtime state)

```
# nginx.env
SSL=1
SSL_CERTS_URL=stashbox/config/ssl.crt
SSL_KEY_URL=stashbox/config/ssl.key
```

```
# syncro.env
SYNCHRO_CONFIG_URL=stashbox/config/config.json
```

```
# stashbox.env
STASHBOX__mounts__0__mount=/config
STASHBOX__mounts__0__provider=manta
STASHBOX__mounts__0__basePath=~~/stor/api_synchro_io/config
STASHBOX__mounts__0__url=https://us-east.manta.joyent.com
STASHBOX__mounts__0__user=synchro
STASHBOX__mounts__0__keyId=<key id>
STASHBOX__mounts__0__key64=<base64 encoded key>
```

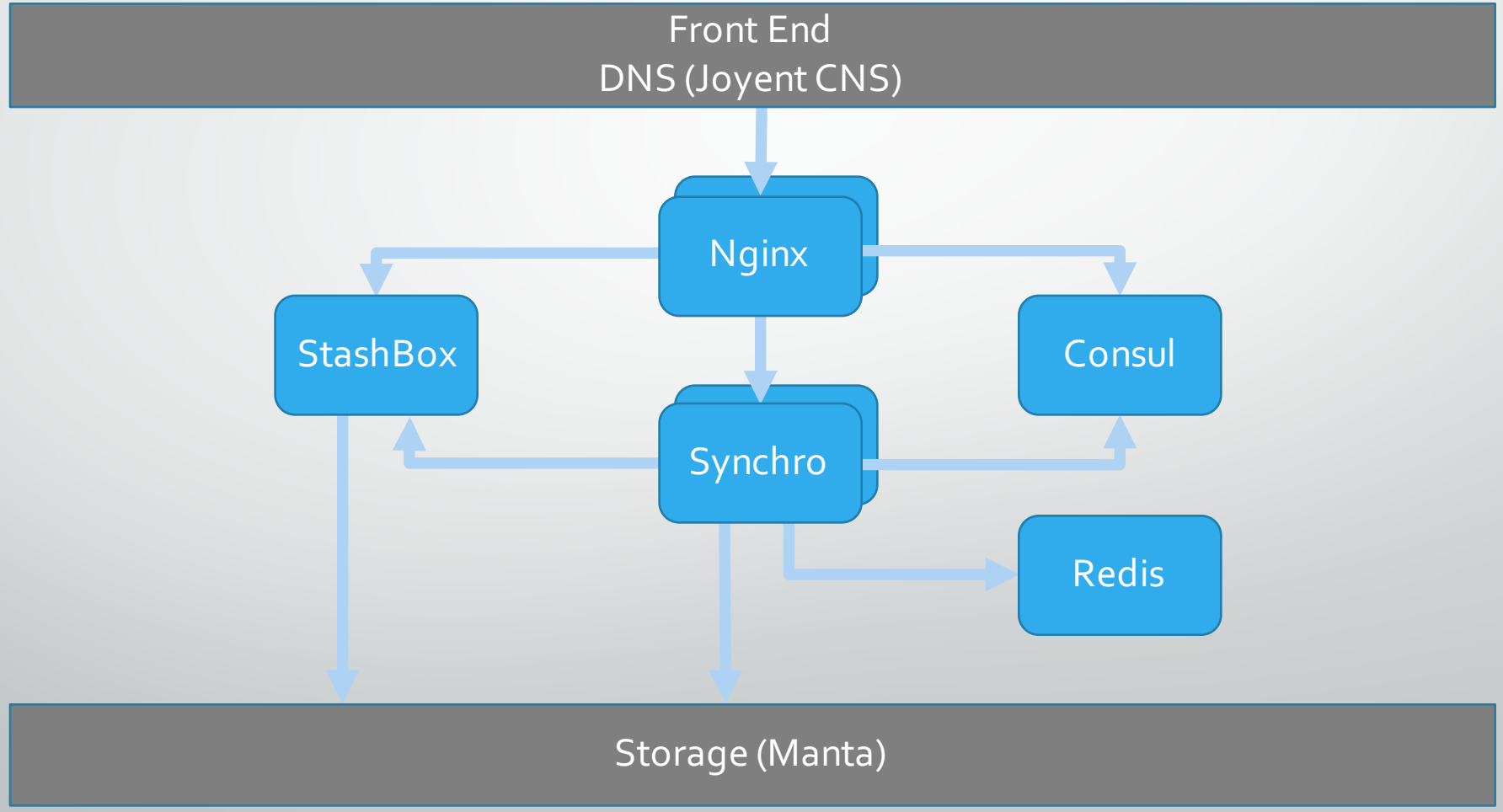
# Nginx preStart with StashBox Support

```
#!/bin/sh
# ContainerPilot preStart
#
preStart()
{
  # Get SSL certs from env or remote URL...
  : ${SSL_CERTS_PATH:="/etc/ssl/certs/ssl.crt"}
  if [ -n "$SSL_CERTS_BASE64" ]; then
    echo $SSL_CERTS_BASE64 | base64 -d > $SSL_CERTS_PATH
  elif [ -n "$SSL_CERTS_URL" ]; then
    curl $SSL_CERTS_URL -s -S -f -o $SSL_CERTS_PATH
  fi;

  # >>> Process ssl.key, nginx.conf.ctmpl as above

  consul-template \
    -once -consul consul:8500 \
    -template "/etc/containerpilot/nginx.conf.ctmpl:/etc/nginx/nginx.conf”
}
```

# Synchro in a Micro-Services Environment (Joyent)



# The Synchro AutoPilot Solution

https://github.com/SynchroLabs/SynchroAutoPilot

- docker-compose.yml
  - synchro/synchro_nginx_ap
  - synchro/synchro_ap
  - synchro/stashbox
  - redis
  - progruim/consul
- Dockerfile w/ ContainerPilot support
  - synchro_nginx_ap
  - synchro_ap
- Real-world nginx.conf.ctmpl
- This presentation (/docs)

# Synchro Production API Server

- Deployed on Joyent Public Cloud (Triton / Manta)
  - Performance is much better
  - Price is lower (even running 5 micro-service containers)
  - Support is outstanding
- Using docker-compose.yml from SyncroAutoPilot GitHub project and public images available from Docker Hub
- Configured at runtime by pointing Nginx and Synchro to StashBox, and pointing StashBox to Manta
  - All configuration, including SSL credentials, API keys, Synchro config, and Synchro apps, are stored on Manta
- All of our application state is in one place, under our control, and portable
- Simulating production on dev is easier and higher fidelity

# Questions

If we don’t get to you, please reach out via email

bob@synchro.io - blake@synchro.io - support@synchro.io

# The End