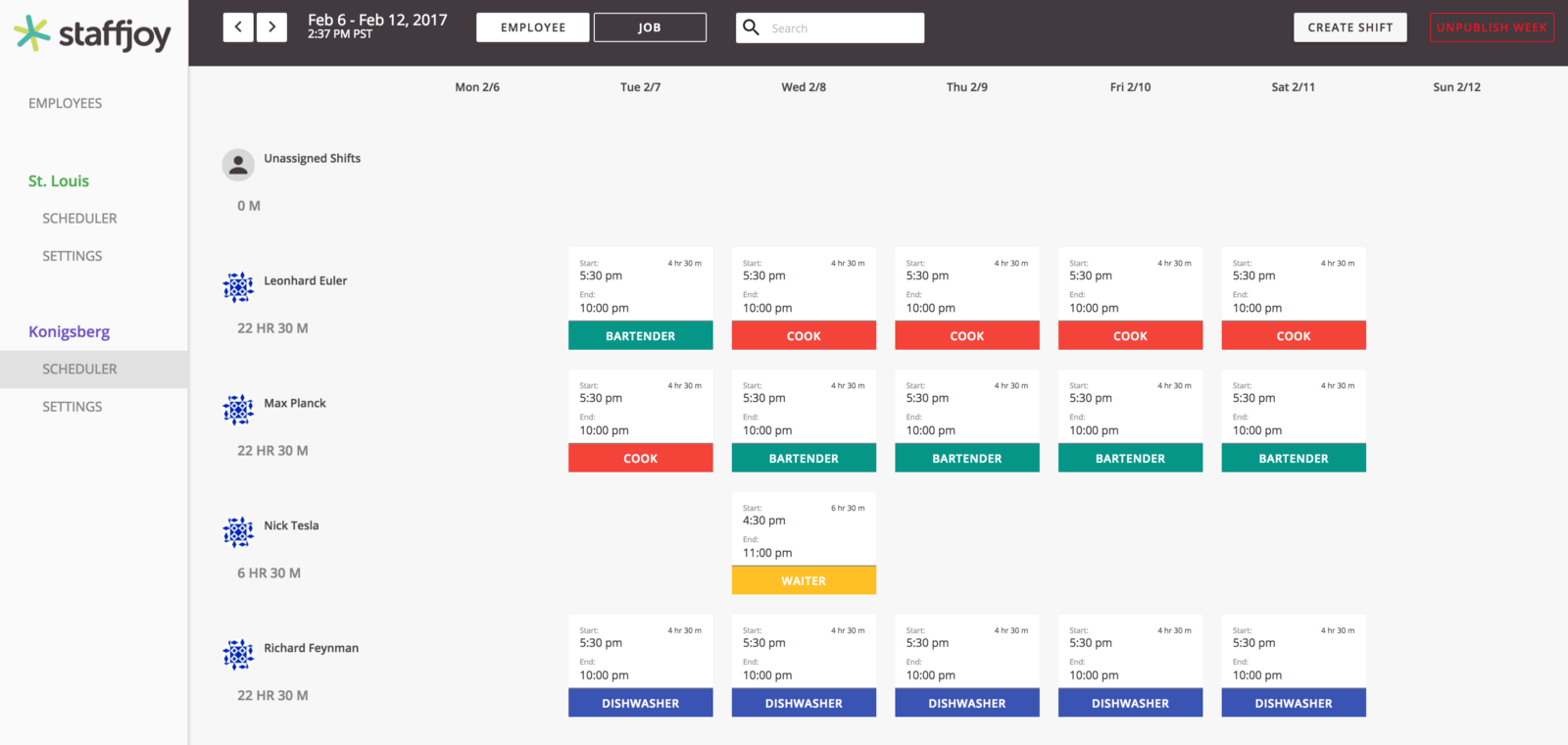
**Staffjoy V2 - README**

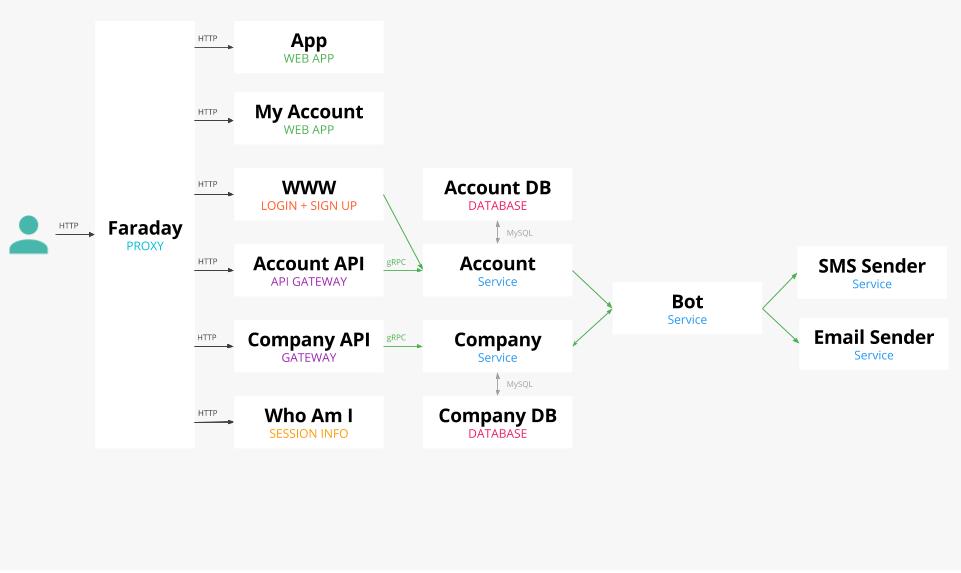
[](https://user-images.githubusercontent.com/1312414/29037396-1f0913ba-7b69-11e7-983f-65bea21718d2.png)

We started building V2 in August 2016, became feature complete in November 2016, and [launched to the press in January 2017](http://venturebeat.com/2017/01/10/staffjoy-raises-1-2-million-to-help-small-businesses-manage-workflow-scheduling/).

This is a *monorepo*, so all of the code for all of the services are in this repo. The core technologies are the [Bazel build system](https://bazel.build/), [Kubernetes](https://kubernetes.io/) (including its DNS for internal service discovery), [Redux](http://redux.js.org/), [Go](https://golang.org/), [Protocol Buffers](https://developers.google.com/protocol-buffers/), [gRPC](http://www.grpc.io/), and [Yarn](https://yarnpkg.com/en/). In staging and production, we used [Google Container Engine](https://cloud.google.com/container-engine/) and their managed databases.

**Services**

[Read about the V2 architecture on our blog](https://blog.staffjoy.com/staffjoys-v2-architecture-9d2fcb4015fd#.pggmlbtmw)

[](https://blog.staffjoy.com/staffjoys-v2-architecture-9d2fcb4015fd#.pggmlbtmw)

* Faraday proxies all traffic from external services to internal ones. It also tells backend services whether a user is logged in. It's the only service that has a public IP address!
* www ([www.staffjoy.com](http://www.staffjoy.com)) is the main marketing website. It also handles login and logout.
* myaccount (myaccount.staffjoy.com) is a single-page javascript app that lets users modify their accounts
* account-gateway (account.staffjoy.com) is the externally-available REST api for modifying accounts. It converts REST to gRPC for the accounts-datastore
* accounts-server is the internal system that processes gRPC calls and stores/retrieves information using the account database.
* company-gateway (company.staffjoy.com) is the externally-available REST api for modifying companies. It converts REST to gRPC for the companys-datastore
* company-server is the internal system that processes gRPC calls and stores/retrieves information using the company database.
* whoami (whoami.staffjoy.com) is a website that sends information about the current web session for easy access in the front-end.
* superpowers (superpowers.staffjoy-v2.local) is a development-only website that lets you gain super user powers across Staffjoy (denoted as "support" flag on user accounts)
* ical (ical.staffjoy-v2.local) is service serving up a worker's shift list through ical

**External API Standards**

* Services should be RESTful JSON over HTTPS
* Serve the spec at /swagger.json
* Use the apidocs package to serve a swagger UI at /ui/

**Dev**

**Getting started**

Welcome to Staffjoy!

We use a **monorepo** that stores all of our code in this single repo. We use Vagrant to run a kubernetes cluster locally on your laptop. This makes it easy to run all of Staffjoy's services.

**Setting up your Gopath**

If you are running Go code, you should [set up your $GOPATH](https://golang.org/doc/install), then clone this repository into the v2.staffjoy.com package:

mkdir -p $GOPATH/src/

git clone git@github.com:Staffjoy/v2.git $GOPATH/src/v2.staffjoy.com/

**One-time dependencies on host machine (laptop)**

* [Virtualbox](https://www.virtualbox.org/)
* [Vagrant](https://www.vagrantup.com/docs/getting-started/), then run vagrant up to boot the dev server.
* Vagrant host manager: vagrant plugin install vagrant-hostmanager
* unison (see below)
* [modd](https://github.com/cortesi/modd) (either from Go source or binary available [here])

**Installing Unison**

Unison syncs files between the host and the VM in a way that preserves simlinks.

Unfortunately, installing it on OSX is a bit of a pain because the version and the compiler must be the same as the one in Vagrant!

First, install OCaml version 3.12 - you may need to say "yes" to some command prompts:

wget https://raw.github.com/ocaml/opam/master/shell/opam\_installer.sh -O - | sh -s /usr/local/bin 3.12.1

Check that ocaml -version is 3.12.1. If it is not, try removing /usr/local/bin/ocaml and running opam switch 3.12.1.

Then, compile unison with this version of ocaml. You may be prompted for your password.

cd /tmp

curl -O https://www.seas.upenn.edu/~bcpierce/unison/download/releases/unison-2.40.102/unison-2.40.102.tar.gz

tar -xvzf unison-2.40.102.tar.gz

cd unison-2.40.102

make UISTYLE=text

sudo cp unison /usr/local/bin

**Running the Environment**

Run make dev. Code will boot and run at [staffjoy-v2.local](http://www.staffjoy-v2.local). Note that the first time you do this could take up to 45 minutes in order to provision the VM!

Changes will trigger an automatic rebuild and redeployment. (Check deployment progress at [kubernetes.staffjoy-v2.local/ui/](http://kubernetes.staffjoy-v2.local/ui/)). End the dev server with control + c (and it will automatically shut off the VM).

**Known Bug on OSX Sierra**: If Vagrant cannot find download the box, run sudo rm /opt/vagrant/embedded/bin/curl. ([Bug tracking link](https://github.com/Varying-Vagrant-Vagrants/VVV/issues/354))

If you run into issues with stuck deployments in development - then run make dev-k8s-fix then re-run make dev.

**Accessing the environment**

Access the VM by running vagrant ssh. Code is located in /home/vagrant/golang/src/v2.staffjoy.com/ (aliased to $STAFFJOY, i.e. cd $STAFFJOY).

To build code and run it locally, in vagrant go to the code directory $STAFFJOY in vagrant, then run make dev-build for a one-time build.

If things are really goofing, run vagrant destroy -f then rebuild.

**Debugging**

* There is a **known bug** where kubernetes does not come back after vagrant halts. To identify this, ssh into vagrant (vagrant ssh), then examine running docker containers (docker ps). If it's not running dozens of contianers, there's a problem. **To fix this:**, run make dev-k8s-fix. (Your local data may be wiped out).
* If files are not syncing between your laptop and vagrant (to Vagrant's $STAFFJOY directory) - try running make dev again, or manually running unison (the syncer) on the host machine with ./vagrant/unison.sh.
* If your machine can't keep up with autobuilding on changes, it may be preferred to stop using make dev and to instead call its two subcommands separately (manually): ./vagrant/unison.sh to sync files on your computer, then SSHing into Vagrant and manually triggering builds (cd $STAFFJOY && make dev-build)

**Development resources**

* [Kubernetes UI](http://kubernetes.staffjoy-v2.local/ui/). We use the development namespace. You can see logs from a "pod" (container) through the UI ([link](http://kubernetes.staffjoy-v2.local/api/v1/proxy/namespaces/kube-system/services/kubernetes-dashboard/#/pod?namespace=development))
* [HTTP Debugger (for faraday)](http://faraday.staffjoy-v2.local/) - use this to examine headers being sent to internal systems
* [Superpowers lets you do magic things](http://superpowers.staffjoy-v2.local) - use it while logged in to get support api permissions in dev.

**Go**

**Assets in Go**

If you are loading assets like templates, CSS, JS, etc - you need to package the the data into the binary. Otherwise, the app will ship and it won't be able to find the assets!

To do this, use the [go-bindata](https://github.com/jteeuwen/go-bindata) project. If you modify any of the asset files, you will need to rebuild them then commit the resulting bindata.go file and commit it. You have been warned!

Most services provide a build.sh file that compiles all the data that needs to be committed.

**Development tools**

The tool [GoConvey](https://github.com/smartystreets/goconvey) is great for seeing tests.

**Environment variables for configuration**

* ENV: Set to development,staging, or production. Null defaults to development
* SENTRY\_DSN: Set to the [Sentry](https://getsentry.com) api key in every Go service for proper error tracking and reporting

**Protocol Buffers**

If you modify the files in protobuf/, run make protobuf to recompile all of the generated files.

⚠️ Please make sure that the version of protobuf matches the runtime version ([see this issue](https://github.com/Staffjoy/v2/issues/5#issuecomment-305704425)):

go get github.com/golang/protobuf/...

cd $GOPATH/src/github.com/golang/protobuf/

# Switch to version that is packaged in app

git checkout df1d3ca07d2d07bba352d5b73c4313b4e2a6203e

# Re-install

go install github.com/golang/protobuf/proto

cd $GOPATH/src/v2.staffjoy.com/

make protobuf

If you're getting started with protocol buffers, here are some resources:

* [Protocol Buffers](https://developers.google.com/protocol-buffers/)
* [gRPC](http://grpc.io)
* [gRPC health checks](https://github.com/grpc/grpc/blob/master/doc/health-checking.md)

**Working Offline**

* Email will break, but you can look at the system logs for the email service to see what would have been sent. (Useful for grabbing account activation links!)
* See all Go documentation installed on the host machie with godoc -http=":8080". You'll be able to see all docs at [localhost:8080](http://localhost:8080/)