SlackOps

Automation Framework

Joe Smith - @Yasumoto

Slack - @SlackHQ

© Slack Technologies, Inc - 2017 - @Yasumoto

Joe Smith

- Operations Engineer at Slack
- Previously at Google and Twitter
- Written mountains of Python
- Run lots and lots of servers- both bare metal and AWS

© Slack Technologies, Inc - 2017 - @Yasumoto

Audience

- Reliability Engineering
- Operations
- DevOps
- SRE (or Production Engineering)

I consider these (and many others!) to have the same thing:

`The desire to build resilient systems`

© Slack Technologies, Inc - 2017 - @Yasumoto

Agenda

1. Running Production Services - Why Automate?
2. Convert a runbook into a python tool
3. Setup your own Ops Repo

© Slack Technologies, Inc - 2017 - @Yasumoto

Approaches to Resiliency

"We will take the site down today!" 💥
— Pretty much no one when they wake up

There are a few universal best practices which we can use to inform how we structure our work.

© Slack Technologies, Inc - 2017 - @Yasumoto

Strategies

- Careful Planning and Procedures
- Extensive Documentation
- Good communication

© Slack Technologies, Inc - 2017 - @Yasumoto

Planning

- Some components are prioritized for speed while others are meant to be canaried and analyzed
- Changes need to be staged to coordinate with each other
- Give teams the tools and visibility they need to make improvements and understand impact
- Identify your rollback strategy **ahead of time**

© Slack Technologies, Inc - 2017 - @Yasumoto

Documentation

- Each code or procedure change should be paired with an update to easily-readable text
- Help your teammates and yourself weeks from now when you need to understand how things work.
- Do not just describe **how** systems are structured, explain *why* they are built that way!
- Additional context can inform future decisions

© Slack Technologies, Inc - 2017 - @Yasumoto

Communication

- Change Management - Coordinating release schedules can be difficult
- Launch Channel - Announce changes, link to more details in a feature-specific Slack channel for the change
- Add links to commits, code reviews, threads in Slack, mailing list posts, and StackOverflow questions
- This enables your team to benefit from the research you've done

© Slack Technologies, Inc - 2017 - @Yasumoto

Review our Strategies

- Careful Planning and Procedures
- Extensive Documentation
- Good communication

© Slack Technologies, Inc - 2017 - @Yasumoto

Growing Pains

- Unexpected changes, forced roll-forwards
- Out of date runbooks
- Missed notifications

© Slack Technologies, Inc - 2017 - @Yasumoto

Good problems to have

Generally, as the team grows, it's no longer possible to understand everything that's happening at once.

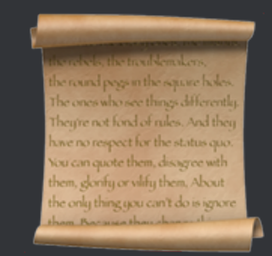
This means the scope of what y'all work on is also increasing!

© Slack Technologies, Inc - 2017 - @Yasumoto

How do tools help with this?[1]

[1] https://davesgeekyideas.com/2016/02/11/thor-hammer-tool-kit/

© Slack Technologies, Inc - 2017 - @Yasumoto

Beyond Runbooks 

Turn a manual checklist into a testable, repeatable set of steps anyone can run

Anytime you discover a sharp edge or workaround, this can be codified in the tool

Reduce sections of "but if this happens, check this dashboard and then do one of three things"

© Slack Technologies, Inc - 2017 - @Yasumoto

Code is readable

"The value of humans is to execute Judgement, the value of computers is to execute instructions"
— Aron, teammate at Slack

Once you have defined the logic for a process, it can be written out in a language for the future

Humans can also read code, but machines can execute explicit instructions.

© Slack Technologies, Inc - 2017 - @Yasumoto

The Tooling Workflow

This process can evolve over a long time and generally improve things.

- One person has all the knowledge in their head 💭
- That person writes down everything they know in a runbook 📜
- Someone sees an annoying or complicated piece and writes a small script to be run instead for a tiny part of the process 💪

The next jump will be the most difficult part!

© Slack Technologies, Inc - 2017 - @Yasumoto

Maintenance

It feels great to have written your first tool!

You may be lucky and have no bugs 🐛

Most likely there are some edge cases- that is okay and expected!

Take some time to figure out what went wrong and how to make things better.

© Slack Technologies, Inc - 2017 - @Yasumoto

The Tooling Workflow

- Later on, another part of the process can be added in and the documentation further updated 💌
- Over time- the runbook becomes "Run this tool we wrote, send bugs to the authors" 💎
- Finally- there is no longer an entry! The tool is run automatically, or the system itself is able to solve that problem 🤖

© Slack Technologies, Inc - 2017 - @Yasumoto

Runbook to Automated Workflow

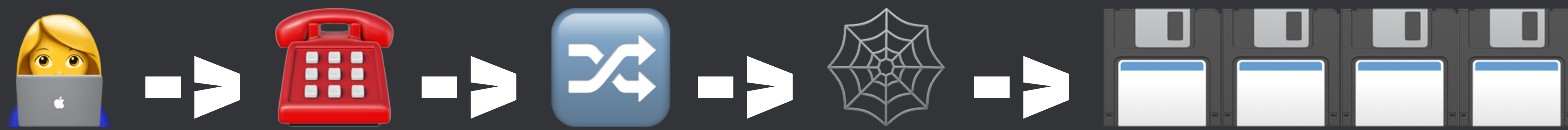
1. Brain 💭
2. Runbook 📜
3. Start of automation 💪
4. Automation evolution (safeguards) 💌
5. Self-contained Tool 💎
6. Fully Automated 🤖

© Slack Technologies, Inc - 2017 - @Yasumoto

What does this look like?

- Let's lay out a very simple application
- Webservice with a database behind it
- Example will be replacing a failed database

© Slack Technologies, Inc - 2017 - @Yasumoto



1. User makes DNS request (Route53)
2. Sent to a load balancer (ELB)
3. Request hits the webserver
4. Queries database (a cluster with one Leader and 3 followers)

© Slack Technologies, Inc - 2017 - @Yasumoto

Replacement

At some point, the inevitable happens. Machine failure.

Our first, trusty database has failed, and is out of service.

We follow the procedures in our runbook.

© Slack Technologies, Inc - 2017 - @Yasumoto

Database Replacement Runbook - Page 1 of 2

1. On all follower hosts, execute **`STOP SLAVE IO_THREAD`**
2. Run **`SHOW PROCESSLIST`** until all hosts show **`Has read all relay log`**
3. On the host that is being promoted, run
   1. **`STOP SLAVE`**
   2. **`RESET MASTER`**

© Slack Technologies, Inc - 2017 - @Yasumoto

Database Replacement Runbook - Page 2 of 2

1. On the hosts that *will not* be promoted, run:
    1. **`STOP SLAVE`**
    2. **`CHANGE MASTER TO MASTER_HOST='<hostname of new leader>'`**
2. Execute on the two followers: **`START SLAVE`**
3. Edit Webserver configuration: Remove the failed database hostname with the hostname of the new leader

© Slack Technologies, Inc - 2017 - @Yasumoto

Progress

So this works well for a while, but we add a few extra clusters, and more machines.

We spend a lot of time keeping up, and make some mistakes

Perilous Step 4: **`CHANGE MASTER TO MASTER_HOST='<hostname'>`**

We may accidentally typo or copy-paste wrong, and all of a sudden the followers think the *wrong host* is leader!

© Slack Technologies, Inc - 2017 - @Yasumoto

Solving this with tooling

First we need to pick a process we understand well so we can automate it.

Changing the leader seems like something we can automate!

Transition into Python for the rest of the talk

© Slack Technologies, Inc - 2017 - @Yasumoto

Building blocks

So.. how do we connect to hosts and execute commands on them from Python?

We need to do two "fundamental" things

- Execute commands on other hosts
  - Use SSH to run commands in a shell
- Find our hosts in the cluster
  - Chef's nodes, puppet's manifests, AWS EC2 API, etc

© Slack Technologies, Inc - 2017 - @Yasumoto

remote_execution.py

```python
from fabric.api import env, run

def execute_command(hostname, command_string, quiet=False):
    """Connect to an external machine, run a command, and return its output.

    This should be used to handle any necessary side effects or preparation for connection.
    """
    env.use_ssh_config = True
    env.host_string = hostname
    try:
        fabric_output = run(command_string, warn_only=True, quiet=quiet)
    except NetworkError as error:
        return ""

    return fabric_output.stdout # Usually want stderr, return code too
```

© Slack Technologies, Inc - 2017 - @Yasumoto

Finding our hosts `mysql.py`

```
from host_management import search
from remote_execution import remote_call

def hosts_in_cluster(cluster_name):
    """Given a cluster name matching our hostname scheme, return all the hosts that belong to a cluster.

    Returns: ["hostnameA", "hostnameB", "hostnameC"]
    """
    return search(cluster_name)

def leader_and_followers(cluster_name):
    """The leader of a cluster is always the lowest-numbered hostname.

    Returns: ("clusterC1", ["clusterC2", "clusterC3", "clusterC4"])
    """
    cluster_hosts = sorted(hosts_in_cluster(cluster_name))
    return cluster_hosts[0], cluster_hosts[1:]
```

© Slack Technologies, Inc - 2017 - @Yasumoto

Picking the new leader - `mysql.py`

```
def change_leader(leader, followers):
    """Update followers to point to the new leader

    Parameters:
        leader: String, like "clusterC1"
        followers: List of Strings, like ["clusterC2", "clusterC3", "clusterC4"]
    """
    for follower in followers:
        print("Pointing {} to follow new leader: {}".format(follower, leader))
        execute_command(follower, "mysql -uadmin \"CHANGE MASTER TO MASTER_HOST='{}'\"".format(leader))
```

© Slack Technologies, Inc - 2017 - @Yasumoto

First Tool 🎉

So now we can build our first tool- details on building it will come later!

```
$ ./promote-mysql-follower --cluster-name=clusterA

Found hosts in clusterA: clusterA2, clusterA3, clusterA4

Pointing clusterA3 to follow clusterA2
Pointing clusterA4 to follow clusterA2
```

© Slack Technologies, Inc - 2017 - @Yasumoto

Better

This helps a bit- we only need to type the new leader's name once, and we automatically detect the followers.

In fact this works out for a few weeks, but the next time we have to perform this process, we accidentally move too quickly.

We did not run **`SHOW PROCESSLIST`** to verify all hosts had caught up to the relay log.

Let's fix that!

© Slack Technologies, Inc - 2017 - @Yasumoto

Detecting caught up relay log `mysql.py`

```
def relay_log_caught_up(cluster_name):
    cluster_hosts = hosts_in_cluster(cluster_name)
    for hostname in cluster_hosts:
        execute_command(hostname, "mysql -uadmin 'STOP SLAVE IO_THREAD'")
    while len(cluster_hosts) > 0:
        for hostname in cluster_hosts:
            output = remote_call(hostname, "mysql -uadmin 'SHOW PROCESSLIST\G'")
            if 'Has read all relay log' in output:
                cluster_hosts.remove(hostname)
    return True
```

© Slack Technologies, Inc - 2017 - @Yasumoto

New Runbook 📕

1. Run `stop-relay-thread --cluster-name=<cluster>`
2. On new leader, run `STOP SLAVE` and `RESET MASTER`
3. `./promote-mysql-follower --cluster-name=<cluster>`
4. Once that has completed, run `START SLAVE` on the two followers
5. Edit to Webserver configuration with leader's

© Slack Technologies, Inc - 2017 - @Yasumoto

Our first issue 💥

Time passes, and unfortunately a teammate runs the script and gets an exception.

```
Traceback (most recent call last):
  File "/home/jmsmith/tools/stop-relay-thread", line 15, in stop-relay-thread
     File "/home/jmsmith/tools/remote_execution", line 25, in remote_execution
SSHConnectionTimeoutError: Could not connect to clusterD4
```

This looks like we were unable to connect to a host!

© Slack Technologies, Inc - 2017 - @Yasumoto

Timeouts and Exceptions

We want to make sure we can be resilient to errors but also *not loop forever*

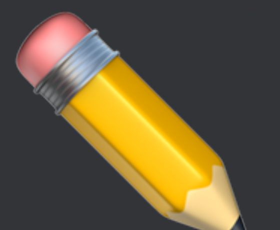
1. Handle **`SSHConnectionTimeoutError`**

2. Keep trying until **`timeout < CONNECTION_TIMEOUT`**

© Slack Technologies, Inc - 2017 - @Yasumoto

```
CONNECTION_TIMEOUT = 300
def relay_log_caught_up(cluster_name):
    cluster_hosts = hosts_in_cluster(cluster_name)
    for hostname in cluster_hosts:
        execute_command(hostname, "mysql -uadmin 'STOP SLAVE IO_THREAD'")
    timeout = 0
    # Continue until we run out of hosts or hit 5min
    while len(cluster_hosts) > 0 and timeout < CONNECTION_TIMEOUT:
        for hostname in cluster_hosts:
            try:
                output = execute_command(hostname, "mysql -uadmin 'SHOW PROCESSLIST\G'")
            except SSHConnectionTimeoutError as error:
                print("Error! Could not connect to {}: {}".format(hostname, error))
            if 'Has read all relay log' in output:
                cluster_hosts.remove(hostname)
            else:
                timeout += 30
                print("Giving {} another {} seconds".format(hostname, CONNECTION_TIMEOUT - timeout))
         time.sleep(30)
    if len(cluster_hosts) > 0:
        print("Warning! Could not connect to {} in time!".format(cluster_hosts))
        return False, cluster_hosts # Let the calling function know which hosts failed
    return True, []
```

© Slack Technologies, Inc - 2017 - @Yasumoto

Full Automation 

Fortunately it turned out the error was indeed temporary.

After a few more weeks, we have some time to finalize the automation!

© Slack Technologies, Inc - 2017 - @Yasumoto

Promote New Leader `mysql.py`

```
def promote_leader(hostname):
  retries = 0
  while retries < 3: # Only one host so we use retries
    retries += 1
    try:
      output = remote_call(hostname, "mysql -uadmin 'STOP SLAVE'; mysql -uadmin 'RESET MASTER'")
      return True
    except SSHConnectionTimeoutError as error: # We could also add a sleep
        print("Error! Could not connect to {}: {}".format(hostname, error))
  return False
```

© Slack Technologies, Inc - 2017 - @Yasumoto

```python
def resume_replication(follower_hostnames):
    """TODO: Add retries + error handling"""
    for hostname in follower_hostnames:
        remote_call(hostname, "mysql -uadmin 'START SLAVE'")
```

© Slack Technologies, Inc - 2017 - @Yasumoto

promote-mysql-follower.py

```
import argparse

from mysql import (leader_and_followers, change_leader, relay_log_caught_up,
    promote_leader, resume_replication)

def main():
    parser = argparse.ArgumentParser(
        description='Replace a downed MySQL Cluster Leader')
    parser.add_argument("--cluster", type=str,
                        help="Cluster Name (such as clusterA)")
    args = parser.parse_args()
    leader, followers = leader_and_followers(args.cluster)
    relay_log_caught_up(args.cluster)
    promote_leader(leader)
    change_leader(leader, followers)
    resume_replication(followers)

if __name__ == '__main__':
    main()
```

© Slack Technologies, Inc - 2017 - @Yasumoto

Final Runbook

1. To replace a database leader, run:
    1. **promote-mysql-follower --cluster=<clusterName>**

© Slack Technologies, Inc - 2017 - @Yasumoto

Process in Code

When there are issues, the code can be reviewed for process changes, git history can be consulted, etc

No more "I forgot that was changed and followed the old process!"

Each time someone submits an improvement or workflow tweak, that will *always* be useful from now on!

© Slack Technologies, Inc - 2017 - @Yasumoto

New procedure 💽

Let's say we also want to make sure we have a current backup in place before we mess around with our databases in the future.

We could write the following:

1. Verify the latest backup is in place
    1. Compare the timestamp of the artifact in S3 with the latest insert of the data
    2. If the backup is not up to date, execute another backup

© Slack Technologies, Inc - 2017 - @Yasumoto

Using `boto3` python module

- AWS Has an Open Source python module named **`boto3`**
- Well respected in the community
- Best way to interact with Amazon Web Services API from Python

© Slack Technologies, Inc - 2017 - @Yasumoto

mysql.py

```python
import time
import boto3

BACKUPS_BUCKET_NAME="tools-backups"
THIRTY_MINUTES = 60 * 30 # 60s per minute, 30 min

def latest_mysql_backups(cluster_name):
    timestamp_filename = "tmp_{}_timestamp.txt".format(cluster_name)
    s3 = boto3.client('s3')
    s3.download_file(BACKUPS_BUCKET_NAME,
                     "{}-backups-timestamp".format(cluster_name),
                     timestamp_filename)
    with open(timestamp_filename, 'r') as fp:
        written = int(fp.read())
        now = time.time()
        if now - THIRTY_MINUTES > written:
            return False
    return True
```

© Slack Technologies, Inc - 2017 - @Yasumoto

Creating a Python Repo[2]

[2] http://massey.dur.ac.uk/training/python/assets/gif/pixel_python.gif

© Slack Technologies, Inc - 2017 - @Yasumoto

Grouping Code

Terms used to describe your layout

- 📝 Module
  - a **`.py`** file which contains python
- 🗂️ Distribution
  - All the files that make up your codebase
  - Can be bundled and released as the same version

© Slack Technologies, Inc - 2017 - @Yasumoto

Multiple Distributions [3]

How do we keep track of not only our code, but the other distrubutions we depend on?

[3] http://www.gifmania.us/Animated-Gifs-Technology/Free-Animations-Computing/Images-Computer-Folders/Yellow-Folder-89637.gif

© Slack Technologies, Inc - 2017 - @Yasumoto

There is a solution based on (and referenced in) PEP 441!

There are some amazing things you can do- please take a look at Brian Wickman's WTF is PEX talk for details!

This is the basis for how we bundle up all of our code into one easy-to-handle tool.

© Slack Technologies, Inc - 2017 - @Yasumoto

PEX

Inside of a properly setup **tools** directory:

```
$ pex -o tools.pex .
$ ./tools.pex
Python 2.7.12
>>> import mysql
>>> mysql.hosts_in_cluster("clusterA")
["clusterA1", "clusterA2", "clusterA3"]
>>>
```

© Slack Technologies, Inc - 2017 - @Yasumoto

Define our Entry Point

Creating a tool as we define it is easy- we just give `pex` a `-c` argument.

```
# note the trailing .
pex -c promote-mysql-follower -o promote-mysql-follower .
```

That gives us the `promote-mysql-follower` file we can then scp around and execute anywhere!

© Slack Technologies, Inc - 2017 - @Yasumoto

Layout

```
$ ls ./tools
bootstrap-python.sh # creates ./.virtualenv
setup.py
build.sh
lib/
test-python.sh # Testing our operational code will come next time!
```

© Slack Technologies, Inc - 2017 - @Yasumoto

Configure the `tools` Distribution

1. What our codebase is called
2. What version of the code we're producing
3. Any links and documentation we have
4. Which tools we produce, called **`console-scripts`**
5. The dependencies we need

© Slack Technologies, Inc - 2017 - @Yasumoto

setup.py

```
"""Distribution Definition for tools module"""
from setuptools import find_packages, setup

setup(name='tools',
      version='0.0.1',
      description='Automation for Operations and Reliability',
      url='https://github.com/Yasumoto/tools',
      packages=find_packages('lib'),
      package_dir={'': 'lib'},
      entry_points={
          'console_scripts': [
              'promote-mysql-follower=tools.bin.promote-mysql-follower:main',
          ],
      }
 )
```

© Slack Technologies, Inc - 2017 - @Yasumoto

Include other Distributions - `setup.py`

```
install_requires=[
    'boto3==1.3.1',
    'fabric==1.11.1',
],
```

© Slack Technologies, Inc - 2017 - @Yasumoto

bootstrap-python.sh

```
#!/bin/sh
if [ $BOOTSTRAP_NEEDED -eq 0 ]; then
    echo Bootstrapping virtualenv
    virtualenv "${REPO_ROOT}/.virtualenv"
    . "${REPO_ROOT}/.virtualenv/bin/activate"
    (./.virtualenv/bin/python2.7" setup.py develop)
    echo 'yup' > "${REPO_ROOT}/.virtualenv/BOOTSTRAPPED"
fi
```

© Slack Technologies, Inc - 2017 - @Yasumoto

Building these tools

We want these accessible to teammates who aren't spending a ton of time writing Python

Also make sure someone in the middle of writing code has a dependable tool available

1. Jenkins job
2. Uploads to S3 (at the Jenkins build number)

© Slack Technologies, Inc - 2017 - @Yasumoto

build.sh

```
#!/bin/sh
# build.sh
python -c '

from __future__ import print_function
    import pkg_resources
    [print(ep) for ep in pkg_resources.iter_entry_points(
        group="console_scripts")]'\

  | grep tools \
  > ./console_scripts.txt
```

© Slack Technologies, Inc - 2017 - @Yasumoto

build.sh

```
#!/bin/sh

pex . -o "./dist/tools.pex"
/usr/local/bin/s3-upload tools-bucket "./dist/tools.pex" $BUILD_NUMBER

while read TOOL; do
  bin=$(echo "$TOOL" | cut -d'=' -f1)
  ep=$(echo "$TOOL" | cut -d'=' -f2)
  "$(dirname "$0")/repex.sh" "./dist/tools.pex" "$ep" "./dist/$bin"
  $SKIP /usr/local/bin/s3-upload tools-bucket "./dist/${bin}" $BUILD_NUMBER
  printf '\033[1;32m%s OK\033[0m\n' "${TOOL}"
done < ./console_scripts.txt
```

© Slack Technologies, Inc - 2017 - @Yasumoto

Deploying these Tools

We have a small helper resource in Chef:

`slackops_tool`

Given a version, pull down the proper binary from S3 after validating its checksum

Enables two types of tools:
1. Human-run
2. Fleet-wide Services (managed by Runit or Monit)

© Slack Technologies, Inc - 2017 - @Yasumoto

```ruby
action :create do
  remote_path = "tools-bucket/tools"
  if new_resource.version
    remote_path = "#{remote_path}/#{new_resource.version}"
  else
    # Pull down latest for environment (prod, dev, or test)
    remote_path = "#{remote_path}/#{node[:environment]}"
  end
  download_path = node[:tool_bucket][:data_home]
  Dir::exist?(download_path) || Dir::mkdir(download_path, 0755)
  s3_file "#{download_path}/#{new_resource.name}" do
    bucket node[:tools][:aws][:bucket]
    remote_path "#{remote_path}/#{new_resource.name}"
    aws_access_key_id node[:tools][:aws][:key]
    aws_secret_access_key node[:tools][:aws][:secret]
    mode new_resource.mode
    owner new_resource.owner
    group new_resource.group
  end
  link "/usr/local/bin/#{new_resource.name}" do
    to "#{download_path}/#{new_resource.name}"
  end
end
```

© Slack Technologies, Inc - 2017 - @Yasumoto

Install a tool

```
tool 'promote-mysql-follower' do
  version 30
end

tool 'monitor-webserver' do
  version 185
  notifies :restart, 'service[monitor-webserver]
end
```

© Slack Technologies, Inc - 2017 - @Yasumoto

Empowers simpler distribution

Once you create all of that, a teammate can:

1. Create a new python tool
2. Add the `console_script` in `setup.py`
3. Kick off a build
4. Install it with config management (like Chef)

© Slack Technologies, Inc - 2017 - @Yasumoto

Configuring for your workplace

- There is a bit of "boiler-plate" that needs to be written once
- Tweaks to suit your needs over time
- Sample layout will be available at https://github.com/Yasumoto/tools
- Additional checks, error handling, and bootstrapping

© Slack Technologies, Inc - 2017 - @Yasumoto

Summary

Operations should move along this trajectory:

1. Institutional Knowledge
2. Documented Runbooks
3. Opinionated and Safe Tools
4. Resilient Self-Healing Systems

https://github.com/Yasumoto/tools

© Slack Technologies, Inc - 2017 - @Yasumoto