ZERO DOWNTIME DEPLOYMENT WITH ANSIBLE

BEKK OPEN

SLIDES & REPO

- http://steinim.github.io/slides/zero-downtime-ansible
- https://github.com/steinim/zero-downtime-ansible

WHAT'S A PROVISIONING FRAMEWORK?

- Automated setup of servers
- Configuration as code

EXAMPLES

- Create users
- Install software
- Generate and manipulate config files
- Start/stop/restart processes
- Set up dependencies between operations

DESCRIBE WHAT TO DO

```bash
#!/bin/bash

if $( command -v vim >/dev/null 2>&1 ); then
  echo "vim is already installed."
else
  apt-get install vim
fi

if $( grep -Fxq "filetype indent off" /etc/vim/vimrc ); then
  echo "set filetype indent off is already in /etc/vim/vimrc."
else
  echo "filetype indent off" >> /etc/vim/vimrc
  # TODO: Do not continue if this fails.
fi

# TODO: Rollback if something fails.
```

DESCRIBE STATE

```
- name: ensure installed vim
  apt: pkg=vim state=installed

- name: set filetype indent off for vim
  lineinfile:
    dest=/etc/vim/vimrc
    line='filetype indent off'
    state=present
```

ANSIBLE

- SSH-based
- Client only (no server)
- YAML configuration
- Push (and pull)
- Supports more than setup and provisioning:
  - Application deployment
  - Remote command execution

BRING UP THE BOXES

```
vagrant up
```

LAYOUT

```
├── ansible.cfg
├── hosts
├── site.yml
├── group_vars
│   └── <group name>
├── host_vars
│   └── <host name>
├── roles
│   ├── <role>
│   │   ├── files
│   │       └── <file>
│   │   └── templates
│   │       └── <template>.j2
│   │   ├── handlers
│   │   │   └── main.yml
│   │   ├── tasks
│   │   │   └── main.yml
```

PLAY!

```
ansible-playbook site.yml
```

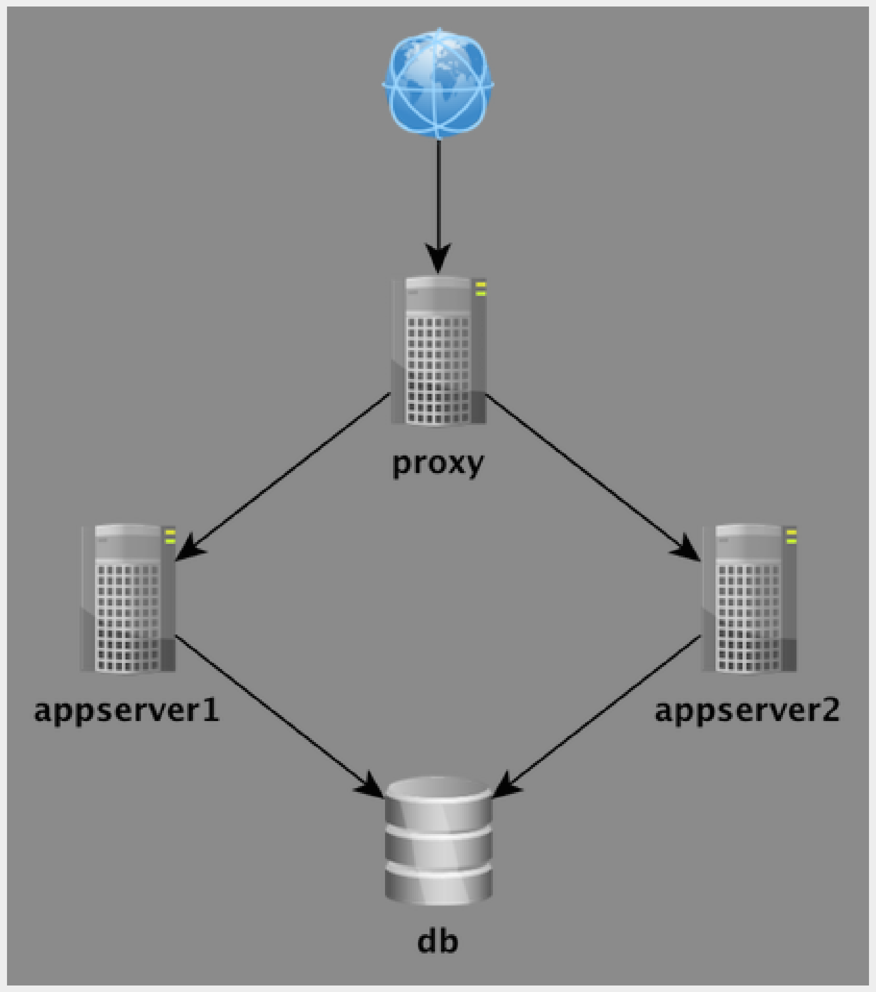
FACTS

- Ansible by default gathers “facts” about the machines under management.
- These facts can be accessed in Playbooks and in templates.

```
ansible -m setup app1.local
```

THE TASK

- An app user 'devops', with:
    - Home directory: /home/devops
    - ssh-key
- A PostgresSQL database.
- Nginx as a reverse proxy.
- An init script installed as a service.
- Deploy an application that uses the provisioned infrastructure.


proxy
appserver1
appserver2
db

HELP!

http://docs.ansible.com/list_of_all_modules.html

TASK1: INSTALL AND CONFIGURE SOFTWARE

```
git checkout start
```

- Modify roles/common/tasks/apt.yml.
- Install Vim.
- Insert the line 'filetype indent off' in /etc/vim/vimrc

```
Help:
http://docs.ansible.com/apt_module.html
http://docs.ansible.com/lineinfile_module.html
```

```
git checkout task1_help
```

TASK1: SOLUTION

```
git diff HEAD origin/task1
git checkout task1 # or keep your own solution
ansible-playbook site.yml --tags apt,vim
```

```
ProTip: Use '--tags', '--skip-tags', '--limit'  and/or 'gather_facts: False'
to reduce execution time.
```

PROGRESS

- Installed software
- Manipulated files

VARIABLES

- Ansible uses variables (a lot!).
- **http://docs.ansible.com/playbooks_variables.html**
  - Inventory
  - group_vars and host_vars
  - Playbook
  - Facts
  - Command line
- Access variables from playbooks: "{{ variable }}"

TASK2: CREATE AN APPLICATION USER

- Create roles/users/tasks/main.yml
- Home directory: /home/devops
- ssh-key
- Use variables! (group_vars)

```
Help:
http://docs.ansible.com/group_module.html
http://docs.ansible.com/user_module.html
http://docs.ansible.com/file_module.html (copy ssh-key)
http://docs.ansible.com/lineinfile_module.html (.ssh/authorized_keys)
http://docs.ansible.com/playbooks_best_practices.html#group-and-host-variables
```

```
git checkout task2_help
```

TASK2: SOLUTION

```
git diff HEAD origin/task2
git checkout task2 # or keep your own solution
ansible-playbook site.yml --limit appservers --skip-tags apt,vim,java
```

```
ssh devops@app1.local
```

PROGRESS

- Installed software
- Manipulated files
- Created a user and set up a ssh-key

TASK3: INSTALL AND CONFIGURE POSTGRESQL

```
roles/postgresql
├── files
|   ├── ACCC4CF8.asc
|   └── postgresql.conf
├── handlers
|   └── main.yml
├── tasks
|   ├── main.yml
|   └── ...
└── templates
    └── pg_hba.conf.j2
```

Use variables (group_vars/all and/or group_vars/dbservers).

Use handler to restart postgresql upon notification

Template: git checkout master -- roles/postgresql/templates/pg_hba.conf.j2

Help:
http://docs.ansible.com/template_module.html (pg_hba.conf.j2)
http://docs.ansible.com/postgresql_user_module.html
http://docs.ansible.com/postgresql_db_module.html
http://docs.ansible.com/playbooks_intro.html#handlers-running-operations-on-change
http://docs.ansible.com/playbooks_best_practices.html#group-and-host-variables

TASK3: SOLUTION

```
git diff HEAD origin/task3
git checkout task3 # or keep your own solution
ansible-playbook site.yml --limit dbservers --tags pg_install
```

```
$ vagrant ssh db
vagrant@db:~$ psql -d devops -U devops -W
devops=> \q
```

PROGRESS

- Installed software
- Manipulated files
- Created a user and set up a ssh-key
- Installed and configured a database and a db user

TASK4: DEPLOY!

```
roles/app
├── files
|   └── init.sh
├── tasks
|   └── main.yml
└── templates
    └── config.properties.j2
```

NB! Use variables (./hosts).

Set 'serial: 1' for appservers in site.yml.

Help:
http://docs.ansible.com/service_module.html

TASK4: SOLUTION

Browse to **http://app1.local:1234/**

```
git diff HEAD origin/task4
git checkout task4 # or keep your own solution
ansible-playbook site.yml --limit appservers --tags deploy
```

WHAT JUST HAPPENED?

```
/home/devops
├── config.properties
├── current -> /home/devops/devops_1416228023.jar
├── previous -> /home/devops/devops_1416221573.jar
├── devops_1416221573.jar
├── devops_1416228023.jar
└── logs
    ├── stderr.log
    └── stdout.log
```

```
/etc/init.d
└── devops
```

PROGRESS

- Installed software
- Manipulated files
- Created a user and set up a ssh-key
- Installed and configured a database and a db user
- Deployed an application to two appservers and enabled it as a service

TASK5: DEPLOY DATABASE

```
roles/db
├── files
|   └── migrate_db.sql
└── tasks
    └── main.yml
```

Help:
http://docs.ansible.com/command_module.html

```
psql -d {{ db.name }} -q -f /tmp/migrate_db.sql
sudo_user: postgres
```

TASK5: SOLUTION

Browse to **http://app1.local:1234/**

```
git diff HEAD origin/task5
git checkout task5 # or keep your own solution
ansible-playbook site.yml --limit dbservers --tags deploy
```

```
$ vagrant ssh db
vagrant@db:~$ psql -d devops -U devops -W
devops=> \dt
devops=> select * from hello;
devops=> \q
```

PROGRESS

- Installed software
- Manipulated files
- Created a user and set up a ssh-key
- Installed and configured a database and a db user
- Deployed an application to two appservers and enabled it as a service
- Migrated the database schema and fetched data from it through the application

TASK6: SET UP PROXY

```
roles/nginx
├── handlers
|   └── main.yml
├── tasks
|   ├── config_nginx.yml
|   ├── install_nginx.yml
|   └── main.yml
└── templates
    └── devops.conf.j2
```

Help:
http://wsgiarea.pocoo.org/jinja/docs/loops.html

TASK6: SOLUTION

Browse to http://proxy.local/ # refresh me many times

```
git diff HEAD origin/task6
git checkout task6 # or keep your own solution
ansible-playbook site.yml --limit proxies --tags nginx
```

PROGRESS

- Installed software
- Manipulated files
- Created a user and set up a ssh-key
- Installed and configured a database and a db user
- Deployed an application to two appservers and enabled it as a service
- Migrated the database schema and fetched data from it through the application
- Set up a reverse proxy for automatic failover between the two appservers

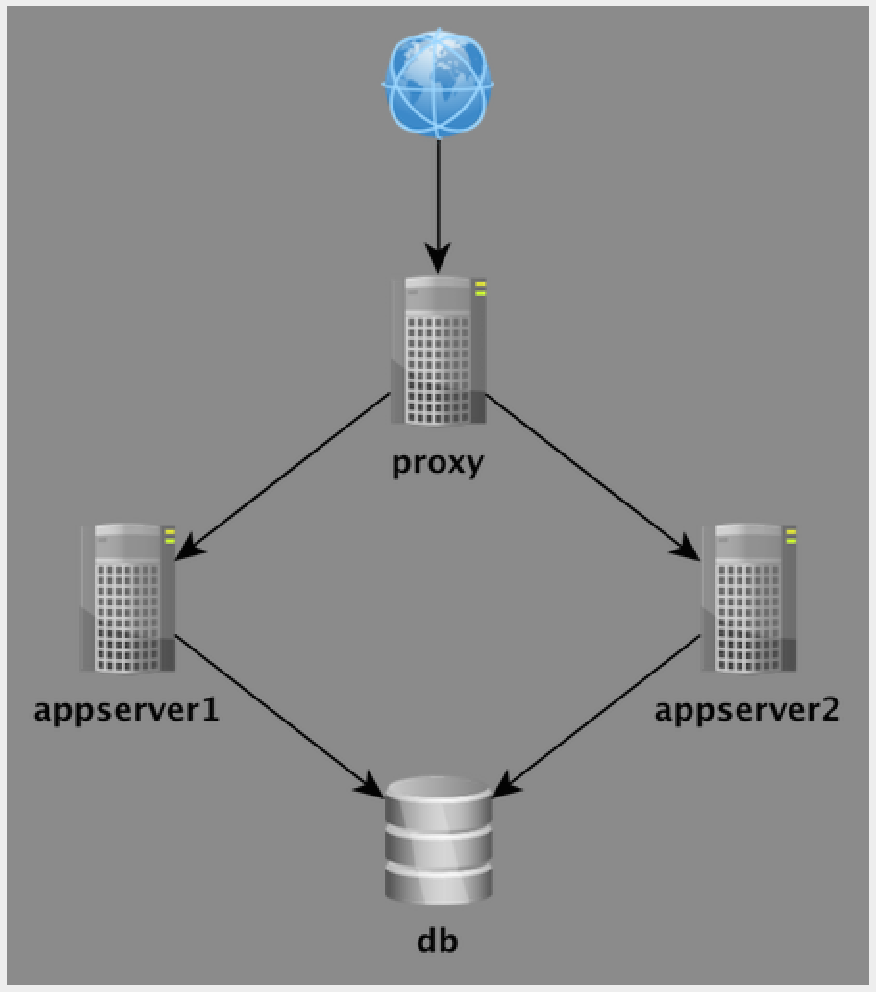
THE EXPAND/CONTRACT PATTERN

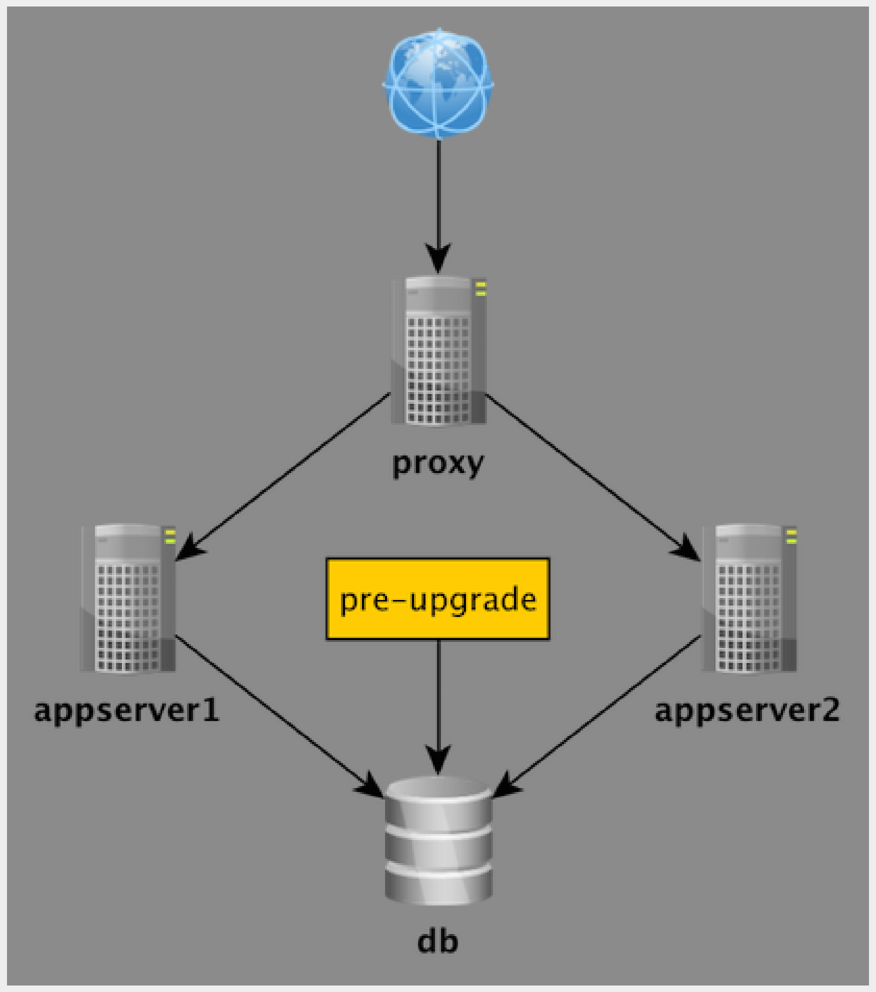
Expand

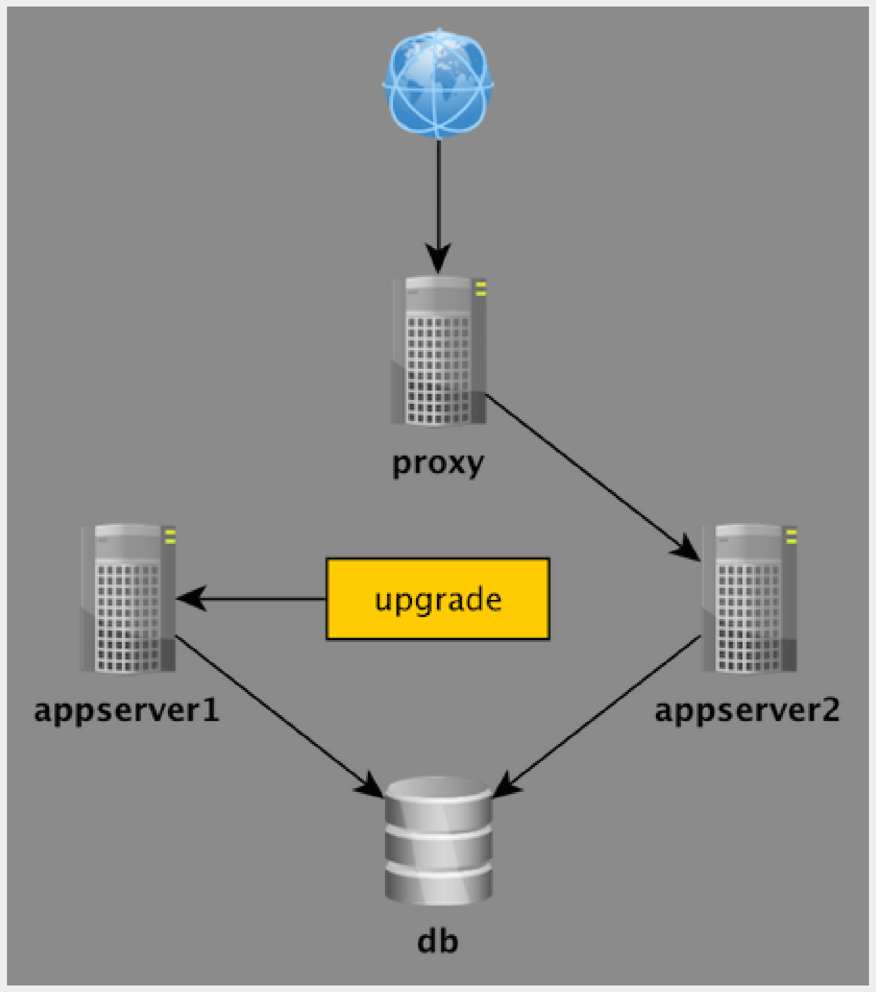
- Add tables
- Add columns
- Tweak indexes

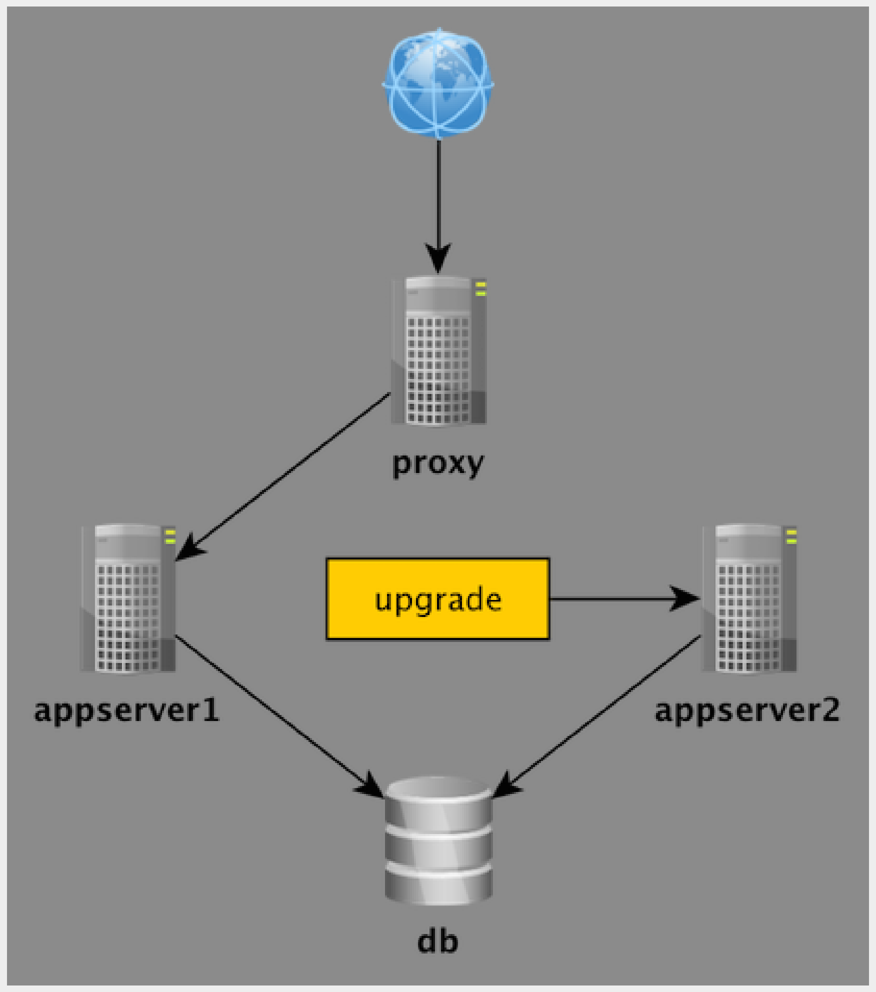
Contract

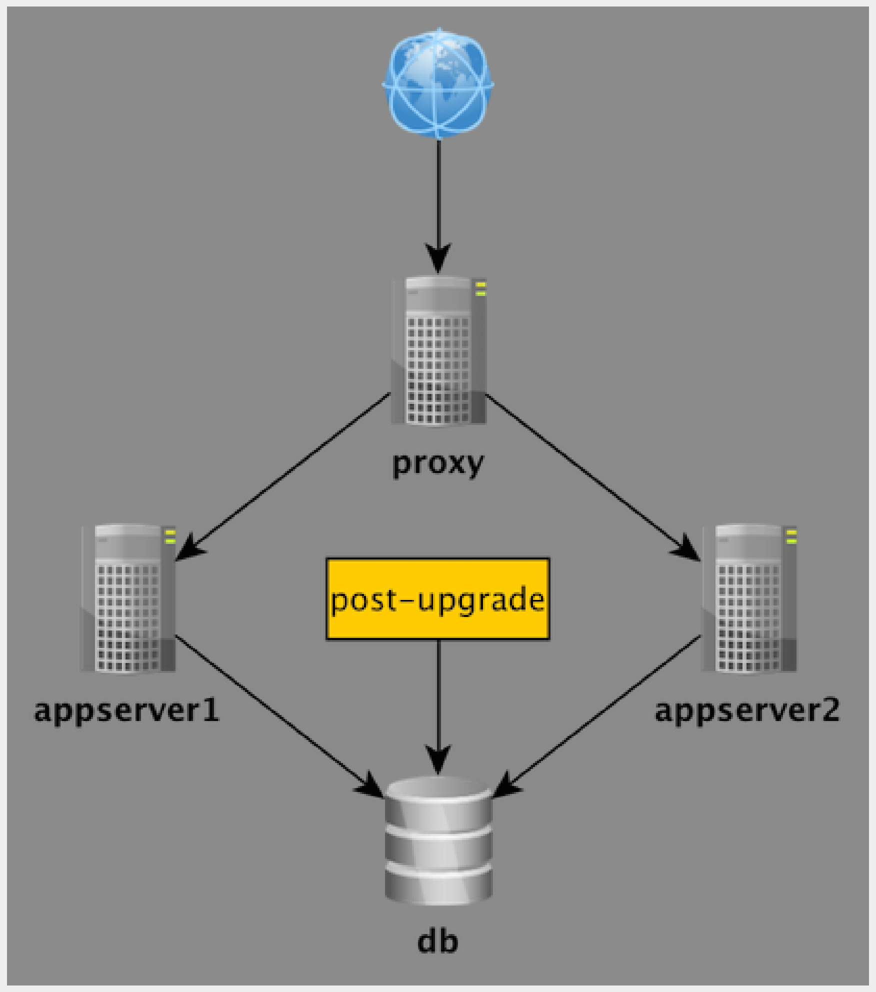
- Remove tables
- Remove columns
- Remove/add constraints

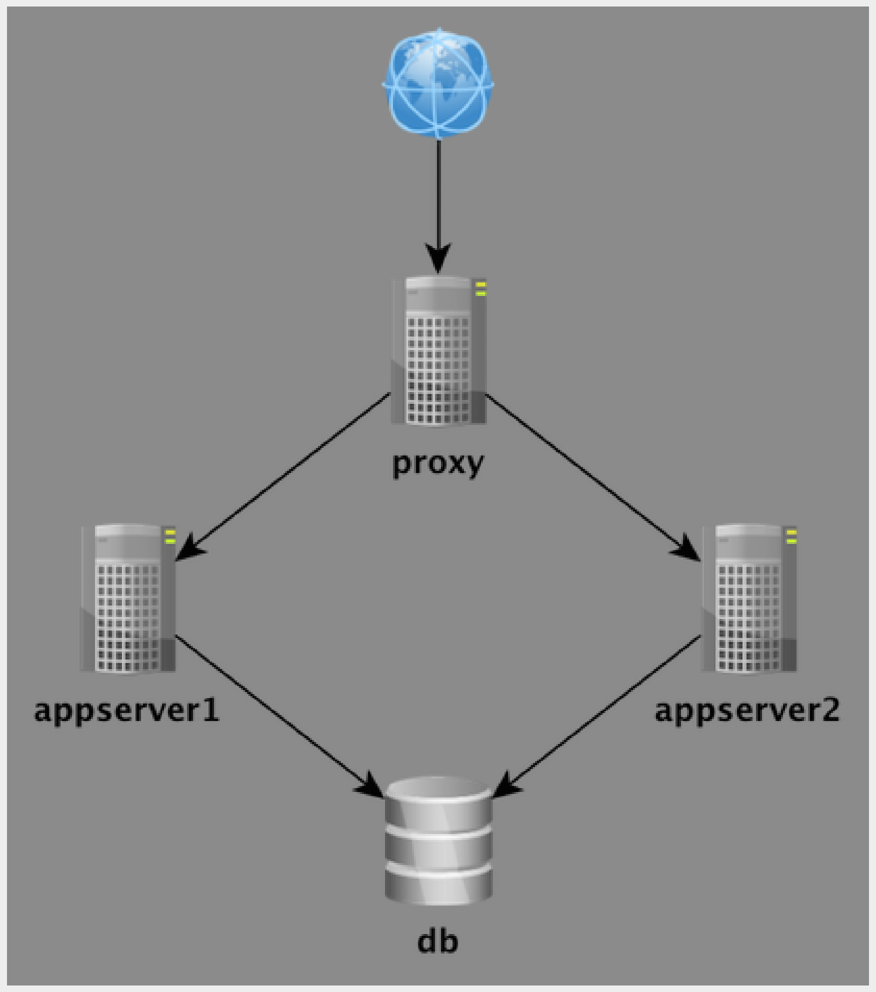

proxy
appserver1
appserver2
db


proxy
pre-upgrade
appserver1
appserver2
db


proxy
upgrade
appserver1
appserver2
db


proxy
appserver1
upgrade
appserver2
db


proxy
appserver1
post-upgrade
appserver2
db


proxy
appserver1
appserver2
db

PLAY TIME :-)

- Suggestions:
  - Change database table name from HELLO to MESSAGES and deploy a new version without downtime.
  - Implement automated rollback.

I HAVE BEEN PLAYING :-)

```
git checkout play
ansible-playbook site.yml --limit appservers,dbservers --tags deploy
ansible-playbook site.yml --limit appservers,dbservers --tags rollback
```

THANK YOU!

@steinim

stein.inge.morisbak@BEKK.no