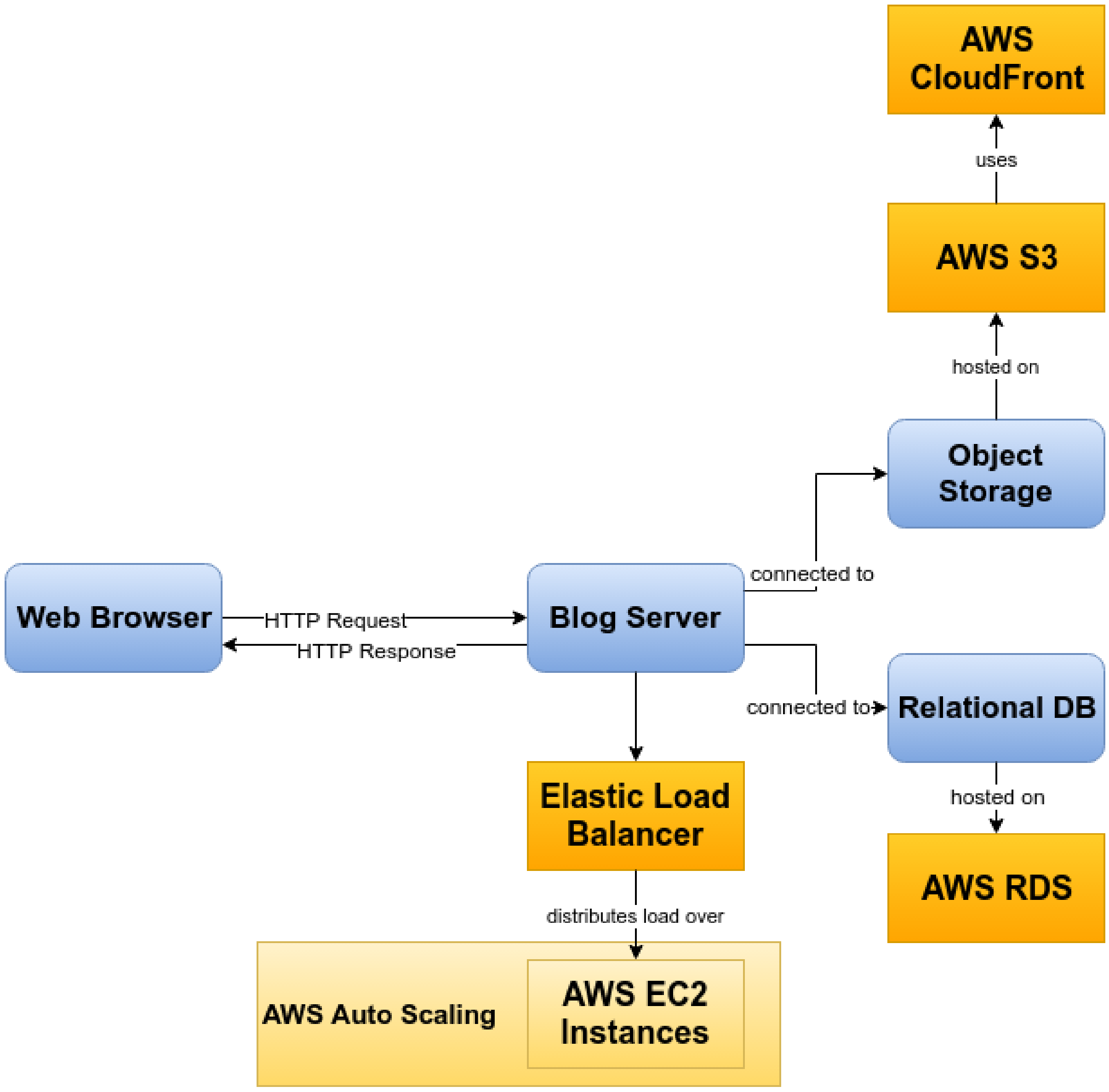
Lab Course SS2016:
Cloud Architectures & Management

Group 4 - Assignment 1 (21.04.2016)
Tobias Freundorfer
Md Rezzakul Haider
Somesh Das

Task 1.2 - Create architecture diagrams for Cloud Applications

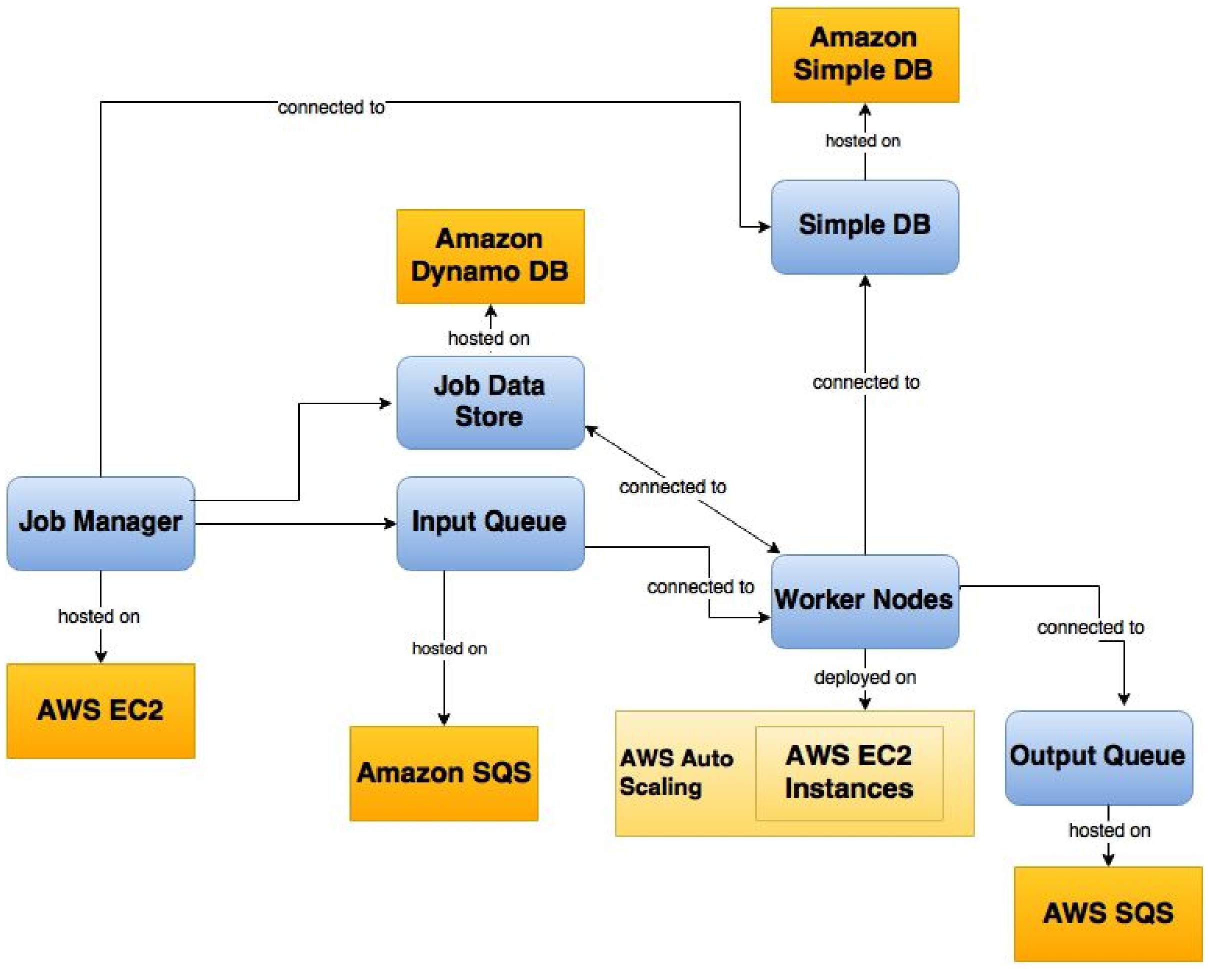
Task 1.2.a - Blog as three-tier Web Application



Requirements:

- Three-tier architecture
- Scaling → AWS Auto Scaling
- Load Balancing → AWS Elastic Load Balancer
- Dynamic Content (Articels) → AWS Relational Database Service
- Static Content (Images, Videos) → AWS S3
- CDN → AWS S3 in combination with AWS CloudFront

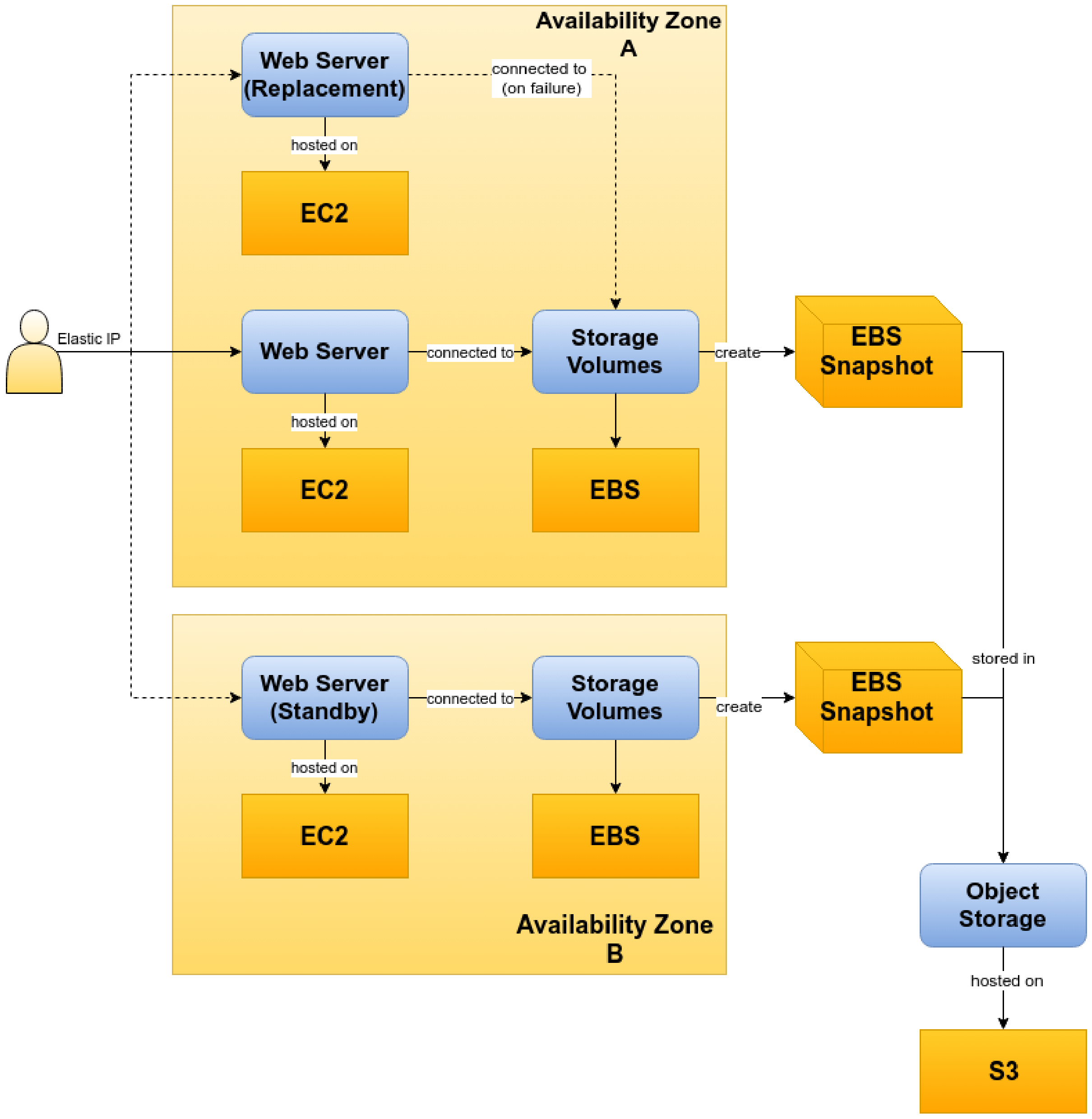
Task 1.2.b - Backend for video sharing platform



Requirements:

- Batch processing
- Job Mnager → Amazon EC2
- Job Data Storage →Amazon Dynamo DB
- Input Queue→ AWS SQS
- Output Queue → AWS SQS
- Worker Nodes → AWS EC2
- Scaling → AWS Auto Scaling

Task 1.2.c - Zero-downtime Web Shop

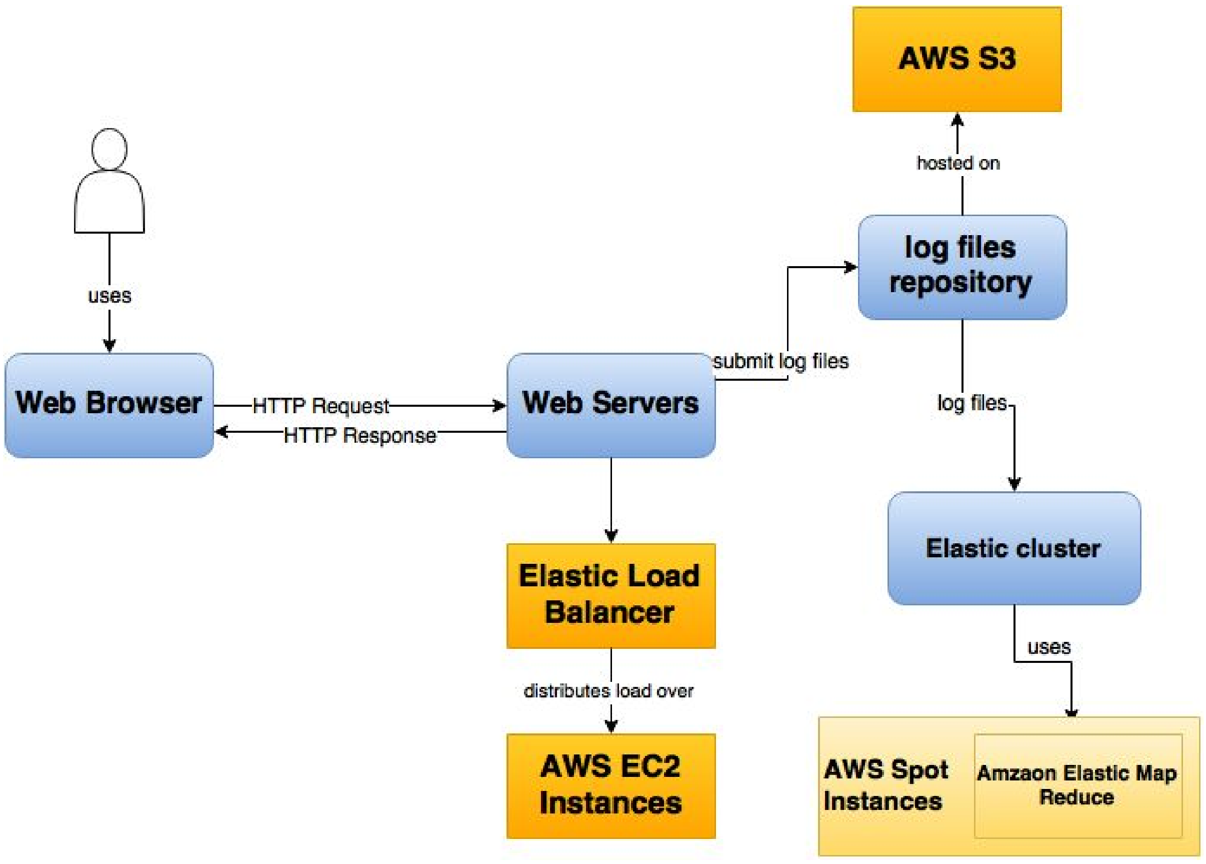


Requirements:

- Fault-tolerance & High availability
  - Redundancy with multiple Availability Zones (AZs)
    - E.g. different AZs on different regions world-wide
  - Elastic IP:
    - Is associated with the AWS Account
    - Enables masking instance or AZ failures programatically → Remapping public IP address to the replacement instance or replacement AZ
  - Replication of the Web Server within a single AZ and dynamic connection to the Amazon Elastic Block Store (EBS)
  - EBS has automatic replication of itself within its AZ

    - Additional storing of snapshots of the EBS to enable recovery and easy copying and sharing of the whole snapshot

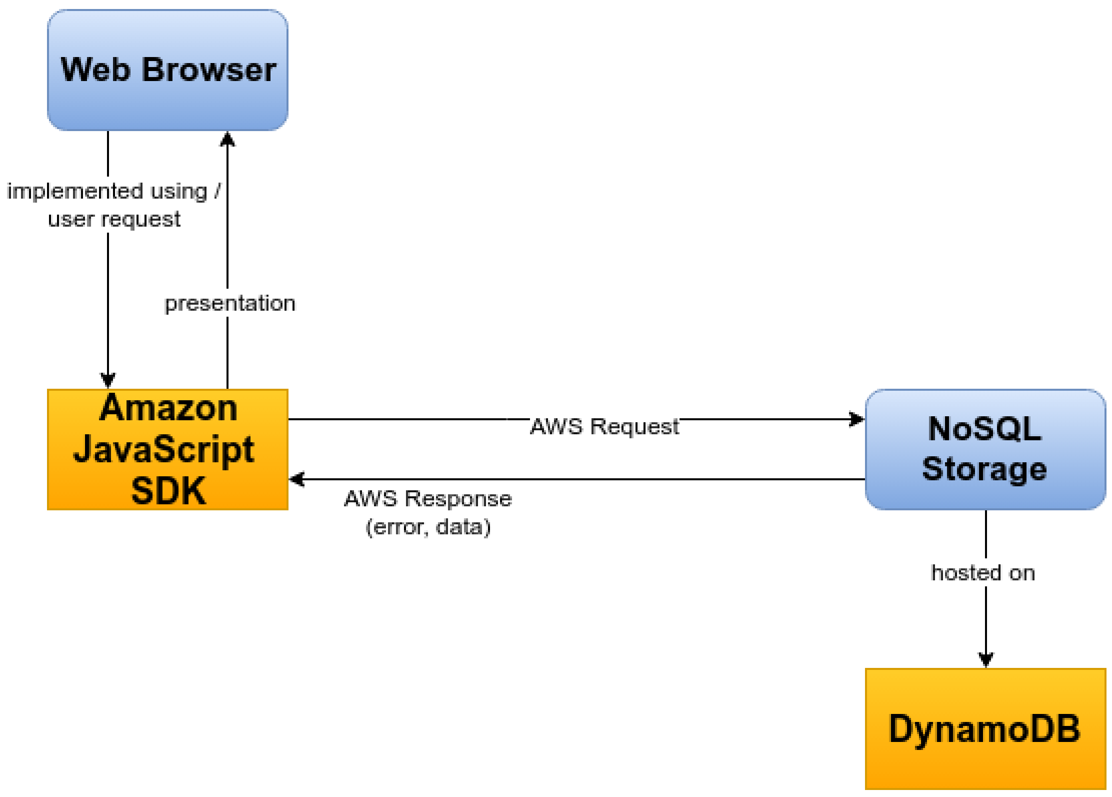
Task 1.2.d -Log analyzer for a Web application



Requirements:

- Log files repository → Amazon S3
- Elastic cluster for logfile processing → Amazon Elastic Map Reduce
- Cost reduction →Amazon Spot Instances

Task 1.2.e - Blog as two-tier Web application



Task 1.3 - Deploy WordPress on single EC2 instance

Initial Setup of AWS:

- Created Virtual Private Cloud (VPC)
- Edited Security Groups
    - Port 22 (ssh with IP address restrictions) 80 (HTTP) 443 (HTTPS)

Connecting via ssh:

- (initially) get according .pem or .ppk File
- chmod 400 on this file
- ssh -i /path/to/key-pair user@host
    - default user for EC2 is: ec2-user

Web Server:

- Check if httpd is running
    - chkconfig --list httpd
      or
    - sudo service httpd status
- document root
    - /var/www/html

Installing WordPress:

- Elastic IP assigned → Else WordPress isn't working anymore because of possible assignment of a new IP address
    - Fix if assignment was forgotten and WordPress doesn't start:
        - Get the CLI for WordPress from: http://wp-cli.org/
        - `curl localhost | grep wp-content` → Get old address
        - Replace old URL with new one
            - `php wp-cli.phar search-replace 'old_site_url' 'new_site_url' --path=/path/to/wordpress/installation --skip-columns=guid`
- **Start MySQL Deamon (if no autostart):**
    - sudo service mysqld start
- **MySQL for WordPress:**
    - username: wordpress
    - password: cloud2016
    - db-name: wordpress-db
- **Wordpress:**
    - username: Admin
    - password: cloud2016
    - For encryption of the user's cookies replace the section in wp-config.php with a generated one
      (using the following generator: https://api.wordpress.org/secret-key/1.1/salt/)

- Example:

```
define('AUTH_KEY',         'PS|7T)aS_4LWu_re=.mr+b#&Fc{2%3-d(9rjVTF=}(kF)Isb@*U7 UQYpQAT~HcC');
define('SECURE_AUTH_KEY',
'mmM%Q,+WZqSm{sQi~@aJj>Fi3p-z:_sd_U9g3+PQ|tS|oSmq6tD[o6&/2;;fin~7');
define('LOGGED_IN_KEY',    '46ZkiR|];O.$&sz|-%+J/2f,i<b~-@Y<T|hU-RYD|B85CgWB_u4*6F+}cgFq+5FC');
define('NONCE_KEY',        'UGO=M#Bi5ceqGj%Ou]E[|J* -:WCVEAD4N7;I~||/QK%1Ee!qZq^=/f8s7+&4:EC');
define('AUTH_SALT',        'q<mD>;w<#Ky>qPWw`)Xtz,Yr5s3]1+?ZfMoN>N5gmN:A=|HT>J;Y1}10f`2co]3z');
define('SECURE_AUTH_SALT', '+S[vk9=giG0&IyvV-o/+M.c5wX={`3(vWZ+XAmg*2z)b%^qA]J{P*.3.]{6J/,DQ');
define('LOGGED_IN_SALT',   '*zMj$+%[K6p2_5.A.(P>CQQcpSW}}6C]Ny_#^6?Jji=|+G.dVW)+LzHB#B (vLC)');
define('NONCE_SALT',       '+&`Nem+F}89L_SKcpN|GKrxe?uC<?NK7=;Xi04ok0t#yKn=5)_~3tce^,{v%v#:9');
```

Wordpress Configuration snippet (wp-config.php) on EC2-Instance:

```
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress-db');

/** MySQL database username */
define('DB_USER', 'wordpress');

/** MySQL database password */
define('DB_PASSWORD', 'cloud2016');

/** MySQL hostname */
define('DB_HOST', ‘localhost');

/** Database Charset to use in creating database tables. */
define('DB_CHARSET', 'utf8');

/** The Database Collate type. Don't change this if in doubt. */
define('DB_COLLATE', '');
```

Task 1.4 - Deploy additional WordPress instances on EC2

Discussion: Now you have multiple WordPress instances. However, each instance has a separate address. What would you have to do to put a loa balancer in front of the WordPress instances, so that all instances can be accessed using a single address?

- When creating an Elastic Load Balancer it receives a public DNS
- Using this address the load balancer is accessible and it will internally (according to health checks and the routing algorithm) route the request to a healthy instance

Discussion: Does it make sense at all to "load-balance" between these WordPress instances? Remember that currently each WordPress instance has its own separate database in the background.

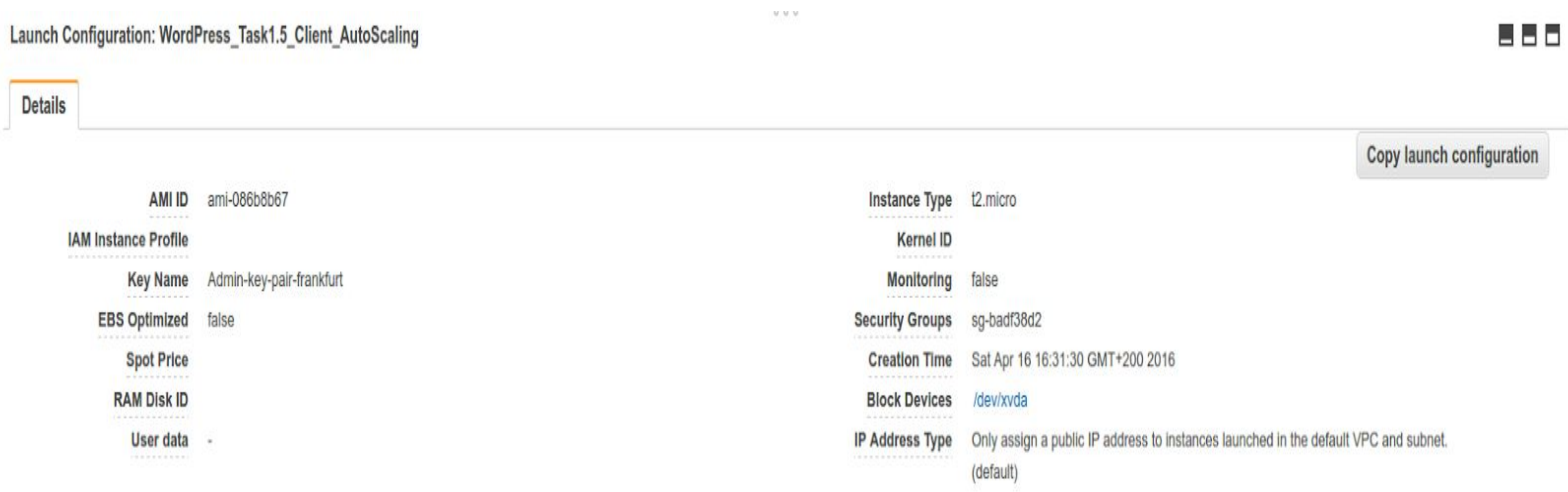
- No because every instance would have data for different users. Therefore no user of the Instance A would be interested in the data lying on Instance B
- If multi-tenancy would be intended both instances should use a shared Database and not their own local

Task 1.5 - Deploy WordPress with Load Balancer

Important: First create the EMPTY DB and then do the installation of WordPress (will also execute the scripts to fill/initialize the DB!!)

Procedure:

1. Create MySQL Database on one EC2-Instance
    a. Configure it for remote access (see following sections)
2. Create WordPress on another EC2-Instance
    a. Configure it to access the shared database (see following sections)
3. Create an AMI from the WordPress instance
4. Define a Launch Configuration using the



Launch Configuration: WordPress_Task1.5_Client_AutoScaling

AMI ID	ami-086b8b67	Instance Type	t2.micro
IAM Instance Profile		Kernel ID	
Key Name	Admin-key-pair-frankfurt	Monitoring	false
EBS Optimized	false	Security Groups	sg-badf38d2
Spot Price		Creation Time	Sat Apr 16 16:31:30 GMT+200 2016
RAM Disk ID		Block Devices	/dev/xvda
User data	-	IP Address Type	Only assign a public IP address to instances launched in the default VPC and subnet. (default)

5. Create an Auto-Scaling Group using the previously defined Launch Configuration

Wordpress Configuration snippet (wp-config.php) on EC2-Instance:

```
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress-db');

/** MySQL database username */
define('DB_USER', 'wordpress');

/** MySQL database password */
define('DB_PASSWORD', 'cloud2016');

/** MySQL hostname */
define('DB_HOST', 'ec2-52-28-132-67.eu-central-1.compute.amazonaws.com:3306');

/** Database Charset to use in creating database tables. */
define('DB_CHARSET', 'utf8');
```

```
/** The Database Collate type. Don't change this if in doubt. */
define('DB_COLLATE', '');
```

MySQL Configuration - on another EC2-Instance:

- Open inbound port for the MySQL database (here Port: 3306)
- Give a user (preferably not root) remote access:
    - `GRANT ALL PRIVILEGES ON *.* TO 'USERNAME'@'%';`
      `FLUSH PRIVILEGES;`
- To ensure the necessary rights are set:
    - `SELECT host FROM mysql.user WHERE User = 'USERNAME';`
- Edit mysql-Deamon (mysqld) config:
    - Open file /etc/my.cnf
        - bind-address='AdressToBindTo'
        - port='PortToBindTo'
            - Do not forget to open this Port in the security group of the MySQL database EC2-instance
    - Restart service
        - sudo service mysqld restart
- Also (as always when configuring WordPress) edit the user cookie encryption according to Task 1.3 - Deploy WordPress on single EC2 instance

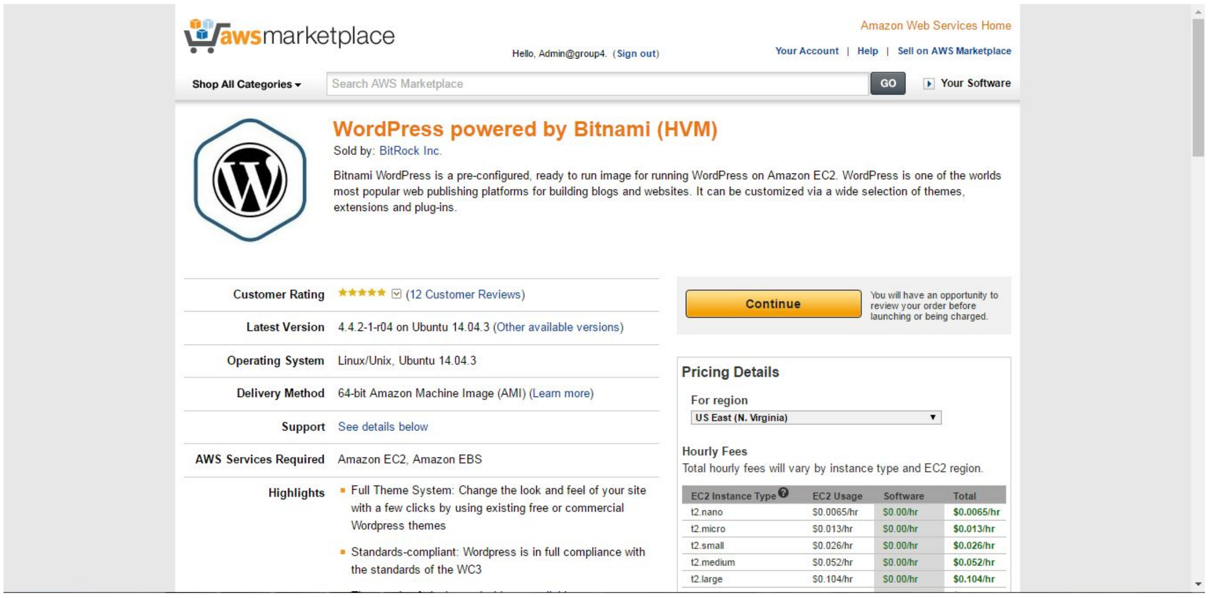
Note:

As far as our team understood the task it was explicitly not intended to do real load balancing (by using for example the Elastic Load Balancer) but only to get a feeling for the idea of the auto-scaling service of AWS. Therefore we decided not to use the Elastic Load Balancer for this task.

Task 1.6 - Deploy WordPress from AMI

Deploy a WordPress instance on EC2 based on a pre-configured AMI (hypervisor-based VM image) :

https://aws.amazon.com/marketplace/pp/B00NN8Y43U/ref=dtl_recsim_B007IP8BKQ_B00NN8Y43U_2/182-2756815-1187550



WordPress powered by Bitnami (HVM)

Sold by: BitRock Inc.

Bitnami WordPress is a pre-configured, ready to run image for running WordPress on Amazon EC2. WordPress is one of the worlds most popular web publishing platforms for building blogs and websites. It can be customized via a wide selection of themes, extensions and plug-ins.

Customer Rating	★★★★★ (12 Customer Reviews)
Latest Version	4.4.2-1-r04 on Ubuntu 14.04.3 (Other available versions)
Operating System	Linux/Unix, Ubuntu 14.04.3
Delivery Method	64-bit Amazon Machine Image (AMI) (Learn more)
Support	See details below
AWS Services Required	Amazon EC2, Amazon EBS
Highlights	Full Theme System: Change the look and feel of your site with a few clicks by using existing free or commercial Wordpress themes Standards-compliant: Wordpress is in full compliance with the standards of the WC3

Continue

You will have an opportunity to review your order before launching or being charged.

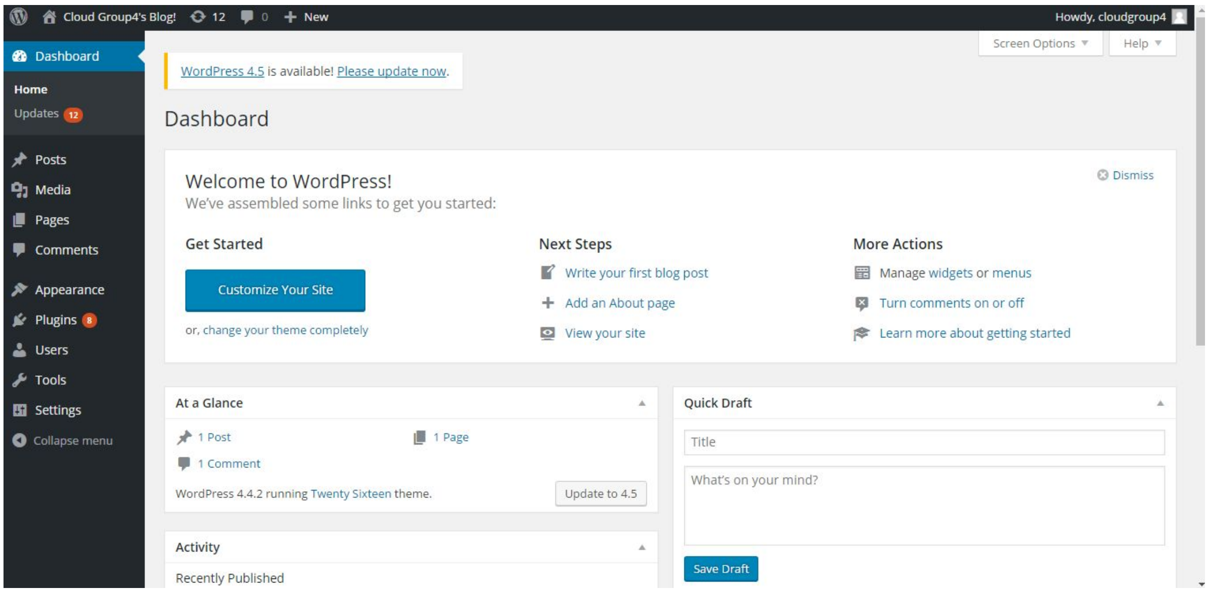
Pricing Details

For region: US East (N. Virginia)

Hourly Fees

Total hourly fees will vary by instance type and EC2 region.

EC2 Instance Type	EC2 Usage	Software	Total
t2.nano	$0.0065/hr	$0.00/hr	$0.0065/hr
t2.micro	$0.013/hr	$0.00/hr	$0.013/hr
t2.small	$0.026/hr	$0.00/hr	$0.026/hr
t2.medium	$0.052/hr	$0.00/hr	$0.052/hr
t2.large	$0.104/hr	$0.00/hr	$0.104/hr



WordPress 4.5 is available! Please update now.

Dashboard

Welcome to WordPress!

We've assembled some links to get you started:

Dismiss

Get Started

Customize Your Site

or, change your theme completely

Next Steps

Write your first blog post

Add an About page

View your site

More Actions

Manage widgets or menus

Turn comments on or off

Learn more about getting started

At a Glance

1 Post

1 Page

1 Comment

WordPress 4.4.2 running Twenty Sixteen theme.

Update to 4.5

Quick Draft

Title

What's on your mind?

Save Draft

Activity

Recently Published

Acessing Admin Page :

- we need to use " /wp-admin/" with the main url.
- To get the password for admin we need to go to the
- AWS instance -> Actions->Instance Settings -> Get System logs
- We can get the password for first time from this system log.

Task 1.7 - Deploy WordPress from Docker image

If having problems with not resolvable packages in apt-get:

- Possible docker problem with resolving IPv6 nameserver
- Go to file “/etc/resolv.conf”
  - Add “nameserver 8.8.8.8”
  - Add “nameserve 129.69.252.252”

Making docker available for local using (without using sudo):

- sudo groupadd docker
- sudo gpasswd -a ${USER} docker ← add local user to docker group
- log out and log in again

Install Docker on EC2:

- sudo yum install -y docker

Pull the images:

- docker pull wordpress
- docker pull mysql

Don’t forget the security groups:

- Always add your IP for ssh tunneling when changing your location
- Open ports if necessary
  - In our example it would be neccessary to open the inbound port 8080 to access the WordPress running in the docker container from the outside

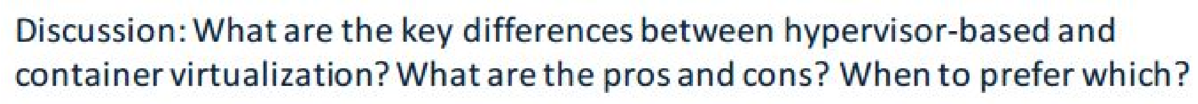
We wrote a short shell script that automates this task:

```
# Pull docker images
docker pull mysql
docker pull wordpress

# Remove ALL existing docker containers (optional but for our repeating tests pretty usefull)
docker rm `docker ps --no-trunc -aq`

# Run mysql database
docker run --name wordpress-mysql -e MYSQL_ROOT_PASSWORD=cloud2016 -d mysql:latest

# Run wordpress
docker run --name wordpress --link wordpress-mysql:mysql -p 8080:80 -d wordpress
```


Discussion: What are the key differences between hypervisor-based and container virtualization? What are the pros and cons? When to prefer which?

Hypervisor	Container
windows system based hypervisor running on underlying physical hardware, We can create another system running on virtual resources and install Linux on it	Basic version of a linux OS containing only necessary functionality.
Hypervisors virtualize on a hardware level	Containers achieve this on an operating system level- by sharing the base operating system's kerne
Hypervisors fail in providing for complete process isolation. As a result, all VM resources are directed to a single process.	Abstract VMs to facilitate isolation of resources to support different processes concurrently. For instance,we can run Arch in one container and Dubian in another at the same time without interfering with each other.
Although simulations are intended to optimize resource utilization, hypervisors largely slow down their servers.	Containers achieve better resource utilization.
Hypervisor is more secure compared to container	Containers are less secure and more vulnerable compared to hypervisors

Task 1.8 - Deploy WordPress using CM tooling (Chef.io)

Basic information:

- Uses code to express infrastructure policies
- Usually uses a centralized Chef Server to download and run the latest chef code → The execution of this code places the system into the desired state
- The chef code, which describes the desired state is also called a 'recipe'
- For the following tasks we will NOT use a Chef Server but only the local-mode
- A recipe is a collection of ressources that describe a particular policy
- A recipe holds the desired state of ressources
    - It declares in which state the ressources should be, but not how to get the ressources into this state. This is abstracted by chef and reduces therefore the complexity
- A ressource represents a piece of infrastructure and its desired state
- Ressources have actions
    - like ':create', ':delete'

    - Using this we can declare for example that a file should be deleted when the recipe gets executed
- A cookbook groups together recipes to make them more managable
- knife is the Chef command line interface (CLI)

Important commands:

- `sudo chef-client --local-mode recipe.rb`
    - Starts the chef client and executes the recipe ‘recipe.rb’
    - Chef does only update the system if the desired state differs from the current state
    - Chef also ensures that if the file was edited (for example manually) that the desired state will be reset when the recipe gets executed again

Important links:

- Get cookbooks from chef.io supermarket
  https://supermarket.chef.io/cookbooks
- Cookbook for WordPress
  https://supermarket.chef.io/cookbooks/wordpress

Chef installation on Ubuntu Server:

- wget https://packages.chef.io/stable/ubuntu/10.04/chef_12.9.38-1_amd64.deb
- sudo dpkg -i <package>
- For our case we also need the Chef Development Kit (for example for repository initialization)
    - wget https://packages.chef.io/stable/ubuntu/12.04/chefdk_0.12.0-1_amd64.deb
    - sudo dpkg -i <package>
- Create repository
    - chef generate repo <repo-name>

Run WordPress cookbook (local-mode):

- Change directory into the created repository
  e.g. here: /var/chef/cookbooks
- `sudo knife cookbook site install wordpress`
    - solves any dependencies to other cookbooks and integrates them into the repository

Configuration file for chef-solo (config.rb):

```
file_cache_path “/home/ubuntu/chef-solo”
cookbook_path “/home/ubuntu/.chef/cookbooks”
```

Declaring the run list items (web.json):

```
{
  “run_list”: [ “recipe[wordpress]” ]
}
```

- Run a solo server using chef-solo:
    - sudo chef-solo -c config.rb -j web.json

What we learned the hard way:

- To use knife install it is mandatory to initialize git within the chef-repo
- Also for every command “**Sudo**” is required !!!
    - We had many problems with HTTP Response Code 403 → This took us quite some time to figure it out

Discussion: How is your CM tooling approach different from the image-based approaches (AMI, Docker)? Pros and cons?

Chef	**Pre-Configured AMI**
Chef recipies are more flexible than custom AMIs.	Specific packages are prebundled instead of installing them after the instance boots.
It’s easier to update.	Control over the timing of package updates to provide a consistent base image for your layer.
It perform updates on running instances.	Booting load balance instances as quickly as possible.
Chef	**Docker**
Chef configurations are packaged into ‘cookbooks’	Docker can build images automatically by reading the instructions from a Dockerfile.
We found lot of code understanding and effort required to make recipes live in chef	Docker containers are build once run anywhere
Chef cookbooks usually have external dependencies.	Docker’s containers are snapshots of a successfully applied configuration

Task 1.9 - Distributed WordPress deployment

Docker Machine:
Is a tool to provision and manage multiple remote docker hosts.

Docker Compose:
Is a tool for defining and running multi-container docker applications.

Note:
Because of time constraints, less team capacity (currently only 2 of 3persons available) and the difficulty to understand and use chef.iof, we couldn’t fulfill the Task 1.9 in time.
If it is possible we would try to submit this task with the submission of Task 2.x or within the timespan of Task 2.x.