[Introduction to our Cloud 2](#_Toc4473)

[Overview 2](#_Toc4474)

[Architecture 2](#_Toc4475)

[Control panel 3](#_Toc4476)

[App controller 3](#_Toc4477)

[Services API 3](#_Toc4478)

[Docker registry 3](#_Toc4479)

[Runners 3](#_Toc4480)

[Loadbalancer 3](#_Toc4481)

[Deployment process 4](#_Toc4482)

[Assignments 5](#_Toc4483)

[1. Migrate a website between regions 5](#_Toc4484)

[2. Deployment with database migrations 5](#_Toc4485)

[3. Optimize demo deployment 6](#_Toc4486)

As part of the technical interview, we would like to let you prepare three different assignments and then discuss them together.

This document contains, in the first section, a short overview of the architecture of Cloud with enough information to start working on the assignments, described in the second section.

While preparing the assignments, please consider that we are more interested in the process and the reasoning behind your solution rather than its technical correctness.

# Introduction to Cloud

## Overview

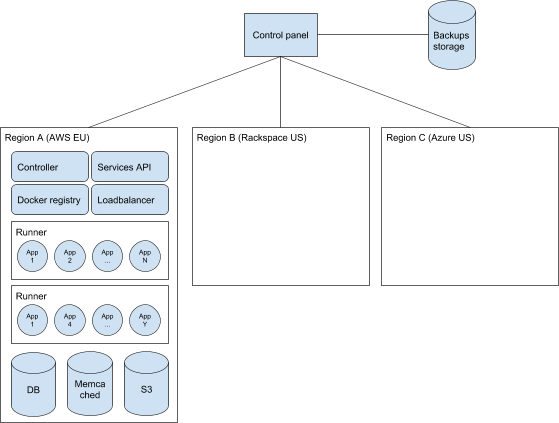
Cloud runs across different cloud service providers, and profits from IaaS offers (e.g. AWS EC2, Rackspace,...) and PaaS providers (AWS RDS, RedisLabs, CloudAMQP,...).

We operate different isolated regions to be able to provide regulatory compliance, provide geolocalized hosting, differentiate offerings based on costs, and other business factors. A single control panel instance coordinates which website is deployed on which region, and acts as the main UI for our end users.

## Architecture

This section gives a first high-level overview of the Cloud architecture.

Cloud is composed of different moving parts. These parts interoperate to make possible to setup, deploy and maintain a large amount of websites in an efficient way. In the following subsections we describe these different parts in more detail.



### Control panel

The control panel is the UI exposed to the end user. The control panel knows about all projects and their users. It stores the configuration, defines the service offerings, stores the source code, manages billing, takes backups,... Once a website is deployed, the control panel is not involved in the serving pipeline.

To deploy a website, the control panel interacts with different components over HTTP REST APIs, by selecting the ones from the region where the website has to be deployed to.

We currently manage one single instance of the control panel.

### App controller

The app controller is the core component of a region. It orchestrates the website instances (building, releasing, scaling, health checks,...) and updates the load balancers each time a deployment or a scaling event requires it.

### Services API

The Services API controls the backing services such as relational databases, S3 storage space, ElasticSearch and others. It can be used to provision, deprovision, backup and restore such services.

At present only the control panel uses it, before deploying a new website or to take regular backups.

### Docker registry

The docker registry stores and serves Docker images for each build and release for each website hosted on the region itself.

### Runners

The runners are actually running the website instances and handling incoming requests. They are also used by the app controller to build Docker images for new deployments.

### Loadbalancer

The loadbalancers take care of routing incoming requests to the correct runner and docker instance for the requested website. If a website has more than one instance, they also load-balance the traffic across all instances.

The loadbalancers also take care of metrics collections which is then used for accounting at a later stage (number of requests, response times, consumed bandwidth,...).

The loadbalancers are able to proxy traffic to arbitrary locations, be it a website instance in the region or an external website.

## Deployment process

When deploying a project, the following happens:

1. If requested, the control panel takes a backup of all provisioned backing services.
2. The control panel makes sure all backing services are provisioned.
3. The control panel makes sure all relevant changes (addons, updates,...) are committed to the source code repo and creates a tarball with the source code.
4. The control panel triggers a deployment on the app controller by pushing the source code tarball.
5. The app controller creates a new Docker build image from the source tarball on a runner.
6. The app controller creates a new Docker release image from the build image created in the previous step, and the configured environment variables.
7. The app controller executes the release scripts (database migrations, cleanups, custom scripts).
8. The app controller pushes the build and the release images to the docker registry.
9. The app controller starts the configured number of instances on different runners (the images are pulled if necessary).
10. The app controller updates the loadbalancer with the new upstream endpoints.
11. The app controller stops the running instances belonging to the previous deployment, if any.

It is possible to deploy a website by skipping steps 3-5. In this case the ID of a previous build to adopt has to be passed to the app controller.

# Assignments

Rules:

1. Choose two assignments to prepare out of the first three (we will stick to them).
2. Please carry out these assignments yourself.
3. Take the time you need, 3-4 hours can surely be spent on these tasks.
4. You are authorized to use online resources, books, and other sources of information.
5. You ~~​can​~~ should ask us questions anytime while you prepare the assignments (don’t worry too much about grouping them in a single email).

## 1. Migrate a website between regions

We need to migrate one or more websites from one region to another, by minimizing downtime. Your job is to plan the migration, enhance the platform with features needed to do so (if necessary) and eventually execute it.

The process shall be repeatable and scalable (we already had to migrate >2k websites between regions once, and we get regular requests from clients to migrate their websites across regions).

Please prepare a runbook for the migration, to be handed in during the interview. We will then discuss it and ask you questions about the process and technical aspects of the migration.

Some example questions we might ask: How did you plan the migration (runbooks, scripts, tests,...)? What are the technical issues which you have to pay attention to? Is it possible to migrate websites with zero downtime? What would you do if the builds are not repeatable (hint: they are not)?

## 2. Deployment with database migrations

We currently run release scripts as part of the deployment process, before starting the instances running the newly deployed code. By default we run Django migrations, but the step is customizable to arbitrary scripts.

For this assignment we would like to focus on database migrations (schema alterations, data conversions,...). Please prepare yourself to discuss some aspects of this topic.

Example questions might be: Are there any flaws in our deployment process? Can we achieve zero-downtime deployments with database migrations? What happens when we have a website with multiple instances? How would you, as a customer, implement and deploy a changeset containing database migrations?

## 3. Optimize demo deployment

At ​https://andlyn.com/demo/get-new/​ you can get a new demo of django CMS. As you can experience, getting a new demo website setup is nearly instantaneous. To achieve this, we maintain a pool of already deployed websites and assign them to users when they request a new one. The size of the pool can be adjusted to match the current demand.

All the demos are based on a “master website”. When creating a ​**new**​ demo website, the master website is first cloned, and deployed like any normal project running on Cloud.

When a demo website times out (e.g. after being used for 15 minutes), the ​**data**​ (database and storage) is overwritten with a fresh copy from the master website and then the cloned instance is redeployed according to the process outlined in the previous section.

This process can easily take between 5 and 10 minutes, and we would like to optimize it and make it faster (<1 minute) so that the pool can be resized in a quicker way. To do so, we want to take advantage of a particular property that demo websites present when compared to normal projects hosted on Cloud.

For this assignment we would like you to identify this property and propose an optimized demo creation/deployment process to fulfill the <1 minute goal.

We might ask questions like: Which impact does this change have on the overall architecture? Which components are involved and which ones need adaptations? Can this particular optimization be applied for other processes inside Cloud?