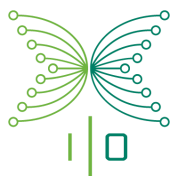
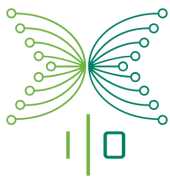


Frontend - System Setup

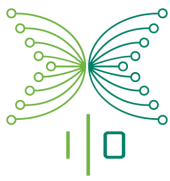




System Requirements	4
Operating System	4
Windows	4
Linux	4
MacOS	4
Node	4
NPM	5
Meteor	5
MongoDB	5
Backend	6
System Configuration	7
Environment Variables	7
Root URL	7
MongoDB URL	7
MongoDB Oplog URL	7
Mailgun Domain Name	8
Mailgun API Key	8
Backend Server IP Address	8
BCC Email Address	8
Backend Client Id	8
Backend Access Token	9
Backend Signature	9
Settings File	9
Sales and Tranches Settings	9
Regional Settings and Legal Documents	11
Legal Documents	13
Price Service Provider	16
Sales App Api Token	18
Backend Settings	19
Loggly	20
AWS	20
Deployment	21
Compose	21
SSL	21
Failover	22
Create a Database	23



Oplog	24
Galaxy	25
Certificate	26
AWS S3	28
Loggly	37
Mailgun	38
Recommendations	39
Example Setup	39
Hosting Services Configuration	39
Galaxy	39
Compose	39
Slack	39
Settings	39
Scripts	45
Known Quick Fixes	45
Scaling Down and Up Container	45
Scale Down	46
Scale Up	47
Restart the Application	48
Stop the containers	48
Start the containers	50



System Requirements

Operating System

Windows

For this project, it is **not** recommended to use a Windows development environment. There are too many issues with tests not running and certain modules not working in Node.js.

Linux

For developing any Linux distribution with a GUI is enough to manage this app. There might be some small issues with rights for the Meteor application, but they can be easily resolved by changing ownership of the .meteor folder. The developers that used Linux are using Ubuntu 16.04.

MacOS

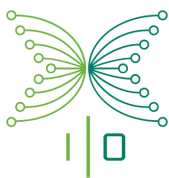
Most of the developers working on this project used Macs although there were some slight problems caused by running wrong versions of Node.js. This can be easily resolved by using Node Version Manager.

Node

The Node current version we're using in this project is 4.6.0. In order to download this version, you can either download it from their web or use Node Version Manager (nvm), following the tutorial.

References:

- Node download: <https://nodejs.org/download/release/v4.6.0/>
- NVM tutorial: <http://stackoverflow.com/questions/7718313/how-to-change-to-an-older-version-of-node-js>



NPM

Node Package Manager is the best tool for importing and updating external packages. The current version in this project is 3.10.8, and it's automatically installed when installing Node, but in case you need to install it manually, you can download it (see the references for the link).

Please make sure to run `meteor npm install --save` once it's installed, so it downloads all the dependencies defined in package.json.

References:

- NPM download: <https://registry.npmjs.org/npm/-/npm-3.10.8.tgz>

Meteor

The application runs in Meteor 1.4.2.3, to install a specific version of Meteor you can do this in two ways:

- You can add the release version as a parameter to the download endpoint

```
curl "https://install.meteor.com/?release=1.4.2.3"
```

- You can use the meteor update command

```
meteor update --release 1.4.2.3
```

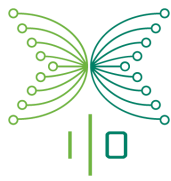
MongoDB

MongoDB is the default database that comes along with Meteor. It's a NoSQL engine which uses JSON standard to store its documents.

MongoDB is installed automatically when Meteor is installed in the project. The current version that is running in this project is 3.2.6.

However, instead of using the latest engine called WiredTiger (which is the default engine in this mongo version), we are using the MMAPv1 engine.

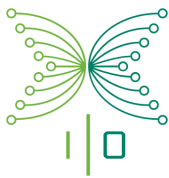
In order to connect to the tests databases, please ensure that the connection to the database uses SCRAM-SHA1 authentication mechanism, and Self-Signed Certificate for the SSL connection protocol.



Backend

Sales process is managed by two independent applications, Sales application that allow users to register, generate orders, see balances, etc; and Backend that handles all Bitcoin related interactions.

This architecture has many security advantages and allows decoupling the order process from the payments itself. Although each application can run independently, both are needed to complete the purchase process. We'll discuss in details about Backend and Sales application configuration in Backend Configuration Settings sections.



System Configuration

Environment Variables

The application is making use of several environment variables, each listed below.

Root URL

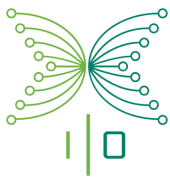
<i>Name</i>	ROOT_URL
<i>Example value</i>	https://*****
<i>Description</i>	Root URL for the application.

MongoDB URL

<i>Name</i>	MONGO_URL
<i>Example value</i>	mongodb://*****@***** *****, *****/*****?ssl =true&authMechanism=SCRAM-SHA-1
<i>Description</i>	URL for MongoDB connection string. Should contain username, password, url and port number for primary and fallback databases, database name, ssl setting, authentication mechanism. Can be found at Compose deployment overview page.

MongoDB Oplog URL

<i>Name</i>	MONGO_OPLOG_URL
<i>Example value</i>	mongodb://oploguser:*****@*** *****/local?authSource=admin&ssl=tr ue
<i>Description</i>	URL for MongoDB oplog. Should contain username, password, url and port number for the oplog, database name associated with the user, SSL setting. Can be found at Compose deployment overview page.



Mailgun Domain Name

<i>Name</i>	MAILGUN_DOMAIN
<i>Example value</i>	*****
<i>Description</i>	Domain name for mailgun. Can be found at Mailgun domain list.

Mailgun API Key

<i>Name</i>	MAILGUN_API_KEY
<i>Example value</i>	key-*****
<i>Description</i>	API key for mailgun. Can be found at Mailgun domain list.

Backend Server IP Address

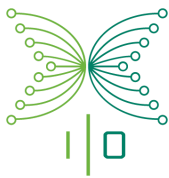
<i>Name</i>	IOHK_BACKEND_SERVER_IP
<i>Example value</i>	https://*****/api/v1/
<i>Description</i>	IP address for IOHK backend server.

BCC Email Address

<i>Name</i>	ATTAIN_BCC_EMAIL
<i>Example value</i>	test-email+test-attain@iohk.io
<i>Description</i>	This is used as a BCC email address when sending out emails.

Backend Client Id

<i>Name</i>	IOHK_BACKEND_CLIENT_ID
<i>Example value</i>	SalesAppTesting
<i>Description</i>	Client ID for IOHK backend server.



Backend Access Token

<i>Name</i>	IOHK_BACKEND_TOKEN
<i>Example value</i>	*****_****_****_****_*****
<i>Description</i>	Token for IOHK backend server.

Backend Signature

<i>Name</i>	IOHK_BACKEND_SIGNATURE
<i>Example value</i>	*****_****_****_****_*****
<i>Description</i>	Signature for IOHK backend server.

Settings File

Sales application settings are divided in two sections. **Settings.json** contains all general and public settings, none of those settings could compromise application security. On the other hand, all sensitive keys and settings are placed in a **Secret.json** for each environment.

Naturally, *production* secret file is not pushed to git repository but it's managed separately and added on deploy by authorised personnel only.

Sales and Tranches Settings

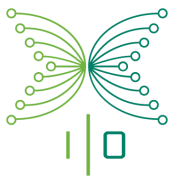
The sales has two main configurations:

- salesStarted: It indicates if the sales are on/off.
- salesLimits: It contains configurations for the tranches.

The “*expireTickets*” configuration is used to manage for how long a ticket is considered valid. It's separated by “bank” and “btc”.

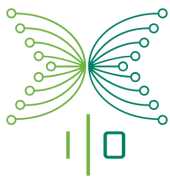
Each period (tranche) has its configuration inside “*salesLimits*”. Here you will have two configurations:

- currentTranche: Indicates which tranche is currently active.
- tranches: Here you can define as many tranches as you want, each one will have two options “*totalAmountAvailable*”⁽¹⁾ and “*overCapacityTolerance*”⁽²⁾



```
// file path: "/config/settings.json"
{
  "public": {
    "salesStarted": true,
    "expireTickets": {
      "bank": {
        "afterInvoiceSent": {
          "days": 3,
          "businessDays": true
        }
      },
    },
    "btc": {
      "afterInvoiceSent": {
        "days": 3,
        "businessDays": false
      }
    }
  },
  "salesLimits": {
    "currentTranche": "t3",
    "tranches": {
      "t1": {},
      "t2": {},
      "t3": {
        "totalAmountAvailable": 14000000,
        "overCapacityTolerance": 0.2
      },
      "t4": {}
    }
  },
  ...
}
```

- (1) totalAmountAvailable: Predefined total of USD equivalent of ADA's.
- (2) overCapacityTolerance: Percentage that the totalAmountAvailable can be exceed.



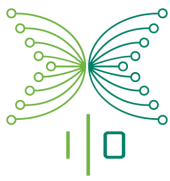
Regional Settings and Legal Documents

Before running the application, it is necessary to define from which residence countries can buyers and distributors enroll from.

You also need to define which countries are allowed for the sale. Buyers from countries where the sale has not started can order ADA, but the ticket is not processed until the sales start.

Example: Available countries in the settings file. ('/config/settings.json')

```
{  
  ...  
  "availableCountries": ["JP", "KH", "CN", "KR", "PH", "TH", "VN"],  
  "residenceCountriesAllowedSale": ["KR"],  
  ...  
}
```

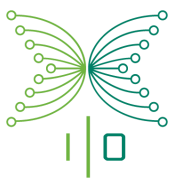


Each available country belongs to a region which must be defined in settings file. Each region has its own configuration related to enrollment and legal documents. For the enrollment section, you need to define the payment option (btc and/or bank) and the “paymentMinimum”⁽¹⁾.

Example: This shows mapping between countries and regions in the settings file.
(`/config/settings.json`)

```
{
  ...
  "regionalMappings": {
    "JP": "regionNippon",
    "KH": "regionCambodia",
    "CN": "regionChina",
    "HK": "regionHongKong",
    "KR": "regionSouthKorea",
    "MY": "regionMalaysia",
    "PH": "regionPhilippines",
    "SG": "regionSingapore",
    "TW": "regionTaiwan",
    "TH": "regionThailand",
    "VN": "regionVietnam"
  },
  ...
}
```

(1) paymentMinimum: This is the minimum amount of USD that a buyer can purchase.



Legal Documents

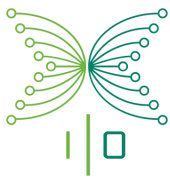
Existing types of policies are: Term of conduct (TOC), PRIVACY and RISK. Each type of user has its own set of documents. Distributor's policies differ depending on their tier.

To add or update policy documents you have to place the documents in *attainenroll/public* folder. Then in the settings.json for each region, you need to write the filename for each policy and describe in what language it is written in (you could define a default option too).

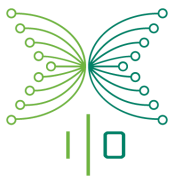
TOC and PRIVACY policies can expire, so the settings needs to have the date of the last update of the files (see "latestPolicyUpdate" in settings file).

Example: Regional settings by each region (settings file '/config/settings.json')

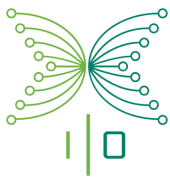
```
{
  ...
  "regionalSettings": {
    "regionNippon": {
      "enrollment": {
        "paymentOptions": ["bank", "btc"],
        "paymentMinimum": 1000
      },
      "latestPolicyUpdate": {
        "buyer": {
          "TOC": "2016-11-26"
        },
        "distributor": {
          "0": "2016-02-04",
          "1": "2016-02-04",
          "2": "2016-02-04",
          "3": "2016-02-04"
        },
        "PRIVACY": "2015-10-01"
      },
      "policies": {
        "TOC": {
          "distributor": {
            "0": [{
              "lang": "ja",
```



```
        "file": "appoiu.pdf"
      }],
      "1": [{
        "lang": "ja",
        "file": "t1k4j3kkjh4l3k4.pdf"
      }],
      "2": [{
        "lang": "ja",
        "file": "t2j3j9dadflk34.pdf"
      }],
      "3": [{
        "lang": "ja",
        "file": "t3y4y2y5uhkjh.pdf"
      }]
    },
    "buyer": [{
      "lang": "ja",
      "file": "toCforbuyerJP_v5a_jp_userpoli.pdf"
    }],
    "default": [{
      "lang": "ja",
      "file": "toCforbuyerJP_v5a_jp_userpoli.pdf"
    }]
  },
  "RISK": {
    "default": [{
      "lang": "ja",
      "file": "riskPolicyJP_v2a_jp_rskpoli.pdf"
    }]
  },
  "PRIVACY": {
    "default": [{
      "lang": "ja",
      "file": "privacy.pdf"
    }]
  }
}
```



```
    },  
    "regionCambodia": {  
      ...  
    },  
    ...  
  }  
}
```

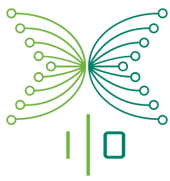


Price Service Provider

In order to get the bitcoin rate price in a controlled manner, Sales application uses a protected endpoint⁽¹⁾ provided by the Backend that gets the rate prices according to the currency code (for example: USD, JPY) from some predefined providers (bitstamp, bitpay, etc). Currently only USD prices are used.

Using Backend as price provider instead of accessing each provider directly, has several advantages:

- Unified format for different providers
- Simplifies provider switching if needed.
- Controlled accessibility, doesn't rely on third party availability.
- Keeps request per minute steady. Some providers had limits on the rpm an IP can query their endpoints. If we query the prices "on demand", there is always a risk of request limit overflow and get the IP blocked.

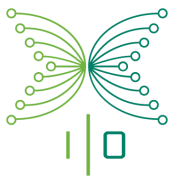


Example: Backend response

```
[
  {
    "providerId": "bitpay",
    "currencyCode": "JPY",
    "price": 112632,
    "lastUpdate": "2016-12-28T12:43:05.705Z"
  },
  {
    "providerId": "kraken",
    "currencyCode": "JPY",
    "price": 114000,
    "lastUpdate": "2016-12-28T12:43:06.319Z"
  },
  {
    "providerId": "blockchain",
    "currencyCode": "JPY",
    "price": 112759,
    "lastUpdate": "2016-12-28T12:43:06.465Z"
  }
]
```

With this answer, the application chooses the first service price that matches these two conditions:

1. It exists in the *Providers ranking*⁽²⁾ (defined in the settings file).
2. Difference between "*lastUpdate*" time and current time does not exceed max outdated tolerance (*outdatedMaxToleranceInMinutes*⁽³⁾).



Settings file (`/config/settings.json`):

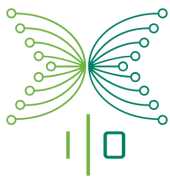
```
{
  ...
  "btcPriceCriteria": {
    "providersRanking": [
      "bitstamp",
      "blockchain",
      "bitpay",
      "bitgo",
      "bitfinex",
      "kraken"
    ],
    "outdatedMaxToleranceInMinutes": 2
  }
}
```

For more information, the algorithm is defined in file: `/server/workers/btc-payment-worker.js`.

- (1) Backend Endpoint *GET*: `"IOHK_BACKEND_SERVER_IP/price/rates/${currencyCode}"`
- (2) Providers ranking: Some providers are more reliable than others, so the application has its own ranking defined.
- (3) `outdatedMaxToleranceInMinutes`: Tolerance criteria defined in the settings file.

Sales App Api Token

Some Sales application API endpoints manipulate sensitive data, and although used by external systems, they are not public. These endpoints uses an static TOKEN based authentication which is predefined in the setting key `SALES_APP_AUTH_API_TOKEN`.

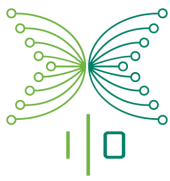


Backend Settings

There are four basic setting configurations needed for the sales-app to communicate with the Backend (Payment processor):

- **IOHK_BACKEND_SERVER_IP:** This is the complete URL to Backend endpoint services root (for example: *https://services.backend.com/api/v1/*)
- **IOHK_BACKEND_CLIENT_ID:** (*) This is used to identify Sales application instance on the backend. On a signed response, this (as a header) will allow the Backend to know who is signing in and easily log the verification process.
- **IOHK_BACKEND_TOKEN:** (*) Every new protected request to the backend needs this Bearer TOKEN header to authenticate. This should be a Read-Write TOKEN.
- **IOHK_BACKEND_SIGNATURE:** (*) This key will be used to sign (encrypt) backend payload responses to yield any middleman interference.

(*) All this tree Keys are generated and provided by the backend itself and linked together, cannot change one without the others. They need to be manually generated and added to both systems to create the bound.



Loggly

The application uses the `miktam:loggly` (version 2.0.0) package for Meteor to communicate with the Loggly service. For this you have to add some settings to the `secret.json` file.

```
"loggly": {
  "enabled" : true,
  "token": "*****_****_****_****_*****",
  "subdomain": "*****.loggly.com",
  "tags" : ["salesapp-test"],
  "auth": {
    "username": "myUsername",
    "password": "myPassword"
  }
}
```

The “enabled” field is not used in the package, but the application uses it as a flag to turn on/off the logs to Loggly.

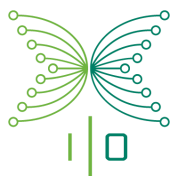
References:

- Miktam:loggly <https://atmospherejs.com/miktam/loggly>

AWS

Amazon S3 is cloud storage for the Internet. Regarding this project, we are currently using AWS to store pictures and documents. To properly configure AWS, you need to set the correct keys under settings file.

```
AWSAccessKeyId: <AWSAccessKeyId>
AWSSecretAccessKey: <AWSSecretAccessKey>
SLINGSHOT_S3_REGION: <SLINGSHOT_S3_REGION>
SLINGSHOT_S3_BUCKET: <SLINGSHOT_S3_BUCKET>
```



Deployment

Deployment is the procedure of moving the application from a local machine to a web server. The steps provided here are for deploying to Galaxy given a Linux machine with Meteor.js installed. In addition a MongoDB instance is created on Compose, and a file container (for user uploaded files) is created on Amazon S3. However the Application can be deployed through any means available for deploying a Meteor application, and the MongoDB instance can be hosted through other providers or being run on the local machine.

These are the requirements for deploying

- An account on Compose (compose.com)
- An account on AWS (aws.amazon.com)
- An account on Galaxy (galaxy.meteor.com)
- The source code
- Linux with Bash installed

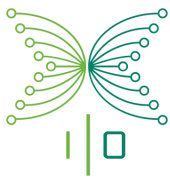
Compose

The application require an instance of MongoDB 3.2, which can be hosted anywhere as long as it's accessible via a URL. The steps in this setup assumes the usage of Compose, but nothing prevents the usage of any other hosting solution. The application needs read and write permission on the database and preferably Oplog access.

SSL

When separating the location of the application and the database, like letting Compose host the database and Galaxy the application, it's important to secure the data link to prevent interception. SSL is a technique to secure this link and should be enabled for the MongoDB deployment. See Environment variables section for more information how to set it up.

SSL has to be enabled when creating a MongoDB 3.2 deployment, and the URL has to be configured in accordance with the Environment variables section.



Failover

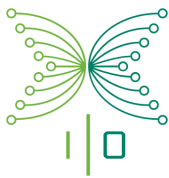
If the database connection goes down the application will normally stop working, so to improve robustness a fallback URL can be used, this technique is called failover.

Failover address is offered by default when setting up an MongoDB 3.2 database, and can be enabled in accordance with the Environment variable section.

The screenshot shows the MongoDB Compose web interface. The top navigation bar includes the 'COMPOSE' logo, a link for 'On-demand backups with the Compose API and Node.js. Read more.', and a 'Sign out' button. The left sidebar contains a menu with icons for Overview, Resources, Browser, Backups, Jobs, Settings, Metrics, Logfiles, Access, Security, and Add-ons. The main content area is titled 'Deployment Overview' and features a green hexagonal icon with a white 'M' and the text 'tangible-mongodb-57'. To the right of the icon is a table of deployment details:

Database	MongoDB 3.2.10
Location	us-east-1 on aws
Status	Healthy
Connection	TCP Available
Usage	2GB of 2GB Disk
Billing	\$53.50/month
Storage Engine	mmapv1
Access	Admin
Notes	Click to add some notes.

Below the deployment overview is a 'Connection info' section. It contains a 'Connection string' field with a text input area. The input text is 'mongodb://<username>:<password>@' followed by a redacted IP address, then '/admin?ssl=true'. The redacted IP address is highlighted with a red rectangle.



Create a Database

The screenshot shows the MongoDB Compose web interface. At the top, there's a navigation bar with the 'COMPOSE' logo, a link to 'On-demand backups with the Compose API and Node.js. Read more.', and a 'Sign out' button. Below the navigation bar is a 'Deployments' section with a 'Create Deployment' button circled in red. The main content area is titled 'New Deployment' and contains a 'New MongoDB Deployment' form. The form has a sidebar on the left with categories: 'Production Deployments' (MongoDB, RethinkDB, Elasticsearch, Redis, PostgreSQL, RabbitMQ, MongoDB (classic)), 'Beta Deployments' (etcd, MySQL, ScyllaDB), and 'Alpha Deployments' (Disque). The 'MongoDB' option is selected. The form fields include: 'Deployment Name' (text input), 'Location' (dropdown menu set to 'US East 1 (Northern Virginia)'), 'Enable SSL access' (checked checkbox), 'WiredTiger Storage Engine' (unchecked checkbox), 'Database Version' (dropdown menu set to '3.2.11'), and 'Allocate initial deployment resources' (slider set to '1GB Storage'). A 'Create Deployment' button is circled in red at the bottom right of the form. To the right of the form, there's a text block: 'Start at \$31.00/month for 3 nodes and 1GB Storage. After that it's just \$18.00/1GB as your usage grows.' and a paragraph about MongoDB's capabilities with a 'Learn more' link.

COMPOSE On-demand backups with the Compose API and Node.js. Read more. → Sign out

Deployments Create Deployment

Don't see the deployment you're looking for? You might need to ask an administrator to grant you access. [Learn more](#) about access and roles at Compose.

New Deployment

New MongoDB Deployment

Production Deployments

- MongoDB
- RethinkDB
- Elasticsearch
- Redis
- PostgreSQL
- RabbitMQ
- MongoDB (classic)

Beta Deployments

- etcd
- MySQL
- ScyllaDB

Alpha Deployments

- Disque

Deployment Name

Location

US East 1 (Northern Virginia)

☒ Enable SSL access

☐ WiredTiger Storage Engine

Database Version

3.2.11

Allocate initial deployment resources

1GB 125GB 1GB Storage

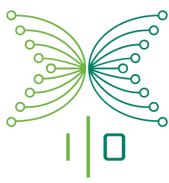
Start with 1GB Storage / 102MB RAM at \$31.00/month.

Create Deployment

Start at \$31.00/month for 3 nodes and 1GB Storage. After that it's just \$18.00/1GB as your usage grows.

MongoDB has led the field of JSON document databases with its powerful indexing and querying, aggregation and wide driver support. It has become the go-to database for many startups and enterprises. [Learn more](#)

Frontend System Setup



Oplog

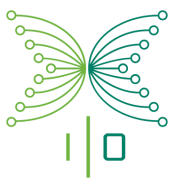
Meteor is built on push notification to update clients with new data changes. When several instances are running in parallel (containers in Galaxy) each connected to a single shared database, the propagation of data between instances is key to keep all instances and clients up to date. Meteor is subscribing to the Oplog to achieve this, and if the Oplog is not configured or goes offline then Meteor will fallback on polling with a frequency of 6 request per minute.

Oplog is not enabled by default in Compose when setting up a MongoDB 3.2 database, instead it has to be manually installed as an Add-on.

The screenshot shows the 'COMPOSE' interface with a sidebar on the left containing icons for Overview, Resources, Browser, Backups, Jobs, Settings, Metrics, Logfiles, Access, Security, and Add-ons (highlighted with a red circle). The main panel displays the 'Deployment Overview' for a MongoDB 3.2.10 instance. The overview includes a green hexagonal icon with a white 'M' and a list of details: Database (MongoDB 3.2.10), Location (us-east-1 on aws), Status (Healthy), Connection (TCP Available), Usage (2GB of 2GB Disk), Billing (\$53.50/month), Storage Engine (mmapv1), Access (Admin), and Notes (Click to add some notes).

The screenshot shows the 'COMPOSE' interface with the 'Add-ons' tab selected in the sidebar. The main panel displays the 'Add-ons' page. On the left, under 'Supported Add-ons', there is a table with three rows: 'Syslog-NG' (Forward logs to Loggly, Papertrail, or Logentries with a Syslog-NG capsule), 'Oplog Access' (Access mongo's oplog via a haproxy portal to the shard's primary), and 'Telegraf' (Forward capsule metrics to Datadog or Librato with a Telegraf capsule). The 'Oplog Access' row is highlighted with a red box, and its 'Configure' button is visible. On the right, there is a section titled 'Add-on Capsules' with text explaining that capsules are easy ways to increase functionality and are attached to the deployment and billed monthly based on hourly usage.

After installation, the Oplog url has to be added in accordance with the Environment variables section.



Galaxy

Regarding scaling, we have experienced some issues when running the app on more than two containers, so use it with caution.

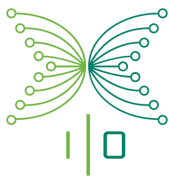
A script, `./bin/deploy/deploy`, has been made available to ease up deployment, for information on how to use it there is a help page available by running `./bin/deploy/deploy --help`. In Galaxy an independent parallel instance is called container.

Galaxy Apps page showing a list of deployed applications. The first application is highlighted with a green flag, indicating it is up and running. The third application is highlighted with a red flag, indicating it is not working.

App Name	Deployed At	Containers	CPU (ECU)	Memory
[Redacted]	February 1st at 9:15pm	3	0.26	18%
[Redacted]	January 12th at 5:45pm	2	0.13	17%
[Redacted]	July 7th at 6:53pm	1	0.52	9%

Galaxy Overview page showing performance metrics. The '3 Containers' metric is highlighted, indicating the current number of containers running.

Metric	Value
Connections	1
CPU (ECU)	0.33
Memory	18%
Containers	3



Certificate


To secure the webpage (login credentials etc), SSL should be used. This is enabled by uploading a certificate via the settings tab. In order to force use of SSL the checkbox “Always use HTTPS” can be checked. Galaxy also has a feature to auto generate a SSL certificate if needed.

To set up a custom certificate follow the instructions below.

You are currently using 3 Standard containers. Scale your app by changing the size of containers now.

DOMAINS & ENCRYPTION

Apps deployed to Galaxy get a unique Galaxy Domain. Add custom domains to make your app accessible via other domain names then configure your DNS provider to point at Galaxy's DNS us-east-1.galaxy-ingress.meteor.com. We recommend that you also enable encryption for your domains to secure sensitive data.


Domains	Encryption	
	Enabled	Primary Domain

+ Add new domain

Domain list ←

DANGER ZONE

Stop app
Stopping an app shuts down all of its containers. When stopped, the app's settings are preserved and organization members may continue to view app logs and performance metrics.


Delete app
Deleting an app shuts down all containers and removes the app from  Galaxy.

STOP **DELETE**

You are currently using 3 Standard containers. Scale your app by changing the size of containers now.

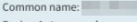
DOMAINS & ENCRYPTION


Apps deployed to Galaxy get a unique Galaxy Domain. Add custom domains to make your app accessible via other domain names then configure your DNS provider to point at Galaxy's DNS us-east-1.galaxy-ingress.meteor.com. We recommend that you also enable encryption for your domains to secure sensitive data.

Domains	Encryption	
	Enabled	Primary Domain

+ Add new domain

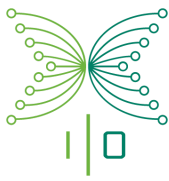
Domain Name test.iohk.io

Automatic Certificate
Common name: 
Expiry: Auto-renewing
Replace with custom certificate **Remove**

Force HTTPS
☐ Always use HTTPS on . When enabled, new incoming clients will transparently redirect to HTTPS. Existing connected clients will not be affected.

CLOSE

Click here to replace the automatically generated certificate for a custom SSL certificate.



Sales App | Frontend - System setup

You are currently using 3 **Standard** containers. Scale your app by changing the size of containers now.

DOMAINS & ENCRYPTION

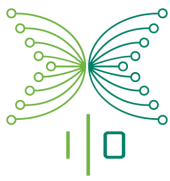
Apps deployed to Galaxy get a unique Galaxy Domain. Add custom domains to make your app accessible via other domain names then configure your DNS provider to point at Galaxy's DNS us-east-1.galaxy-ingress.meteor.com. We recommend that you also enable encryption for your domains to secure sensitive data.

Domains	Encryption	Primary Domain
<div>UPLOAD CERTIFICATE</div> <div>SSL Key & Certificate</div> <div>Key: Choose File Certificate: Choose File</div> <div>Upload here your SSL Key and Certificate</div> <div>CANCEL UPLOAD CERTIFICATE</div>		

+ Add new domain

DANGER ZONE

Stop app

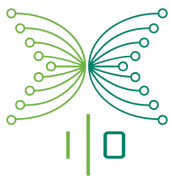


AWS S3

The application requires file storage system to store user's documents. We use AWS S3 as an host.

1. To use AWS S3 you will first need to sign up for AWS. Visit <https://aws.amazon.com/s3/> and press "Sign Up"





2. Choose your account details and press the “Sign in using our secure server”




Sign In or Create an AWS Account

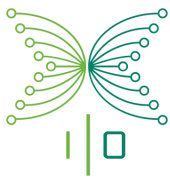
What is your email (phone for mobile accounts)?

E-mail or mobile number:

- ☒ I am a new user.
- ☐ I am a returning user
and my password is:

Sign in using our secure server 

[Forgot your password?](#)



3. Fill in your details and create the account.



Login Credentials

Use the form below to create login credentials that can be used for AWS as well as Amazon.com.

My name is:

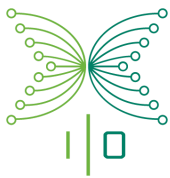
My e-mail address is:

Type it again:

note: this is the e-mail address that we
will use to contact you about your
account

Enter a new password:

Type it again:



4. Fill in your contact information and create the account

Contact Information

☒ **Company Account** ☐ **Personal Account**

** Required Fields*

Full Name*

Company Name*

Country*

United States ▼

Address*

Street, P.O. Box, Company Name, c/o

Apartment, suite, unit, building, floor, etc.

City*

State / Province or Region*

Postal Code*

Phone Number*

Security Check ?



[Refresh Image](#)

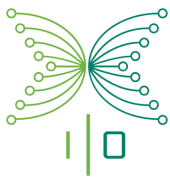
Please type the characters as shown above

AWS Customer Agreement

☐

Check here to indicate that you have read and agree to the terms of the [AWS Customer Agreement](#)

Create Account and Continue



5. Enter your payment information

Contact Information

Payment Information

Identity Verification

Support Plan

Confirmation

Payment Information

Please enter your payment information below. You will be able to try a broad set of AWS products for free via the Free Tier. We will only bill your credit or debit card for usage that is not covered by our Free Tier.

[Frequently Asked Questions](#)

Credit/Debit Card Number

Expiration Date

03 ▼

2017 ▼

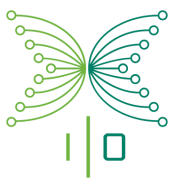
Cardholder's Name

☒ Use my contact address

☐ Use a new address

Continue

6. The next step requires phone verification, please follow the steps presented on screen. You will be called by Amazon to verify.



7. Please choose the appropriate support plan for your needs.

Support Plan

AWS Support offers a selection of plans to meet your needs. All plans provide 24x7 access to customer service, AWS documentation, whitepapers, and support forums. For access to technical support and additional resources to help you plan, deploy, and optimize your AWS environment, we recommend selecting a support plan that best aligns with your AWS usage.

Please Select One

☒ **Basic**

Description: Customer Service for account and billing questions and access to the AWS Community Forums.

Price: Included

☐ **Developer**

Use case: Experimenting with AWS

Description: One primary contact may ask technical questions through Support Center and get a response within 12–24 hours during local business hours.

Price: Starts at \$29/month (scales based on usage)

☐ **Business**

Use case: Production use of AWS

Description: 24x7 support by phone and chat, 1-hour response to urgent support cases, and help with common third-party software. Full access to AWS Trusted Advisor for optimizing your AWS infrastructure, and access to the AWS Support API for automating your support cases and retrieving Trusted Advisor results.

Price: Starts at \$100/month (scales based on usage)

☐ **Enterprise**

Use case: Mission-critical use of AWS

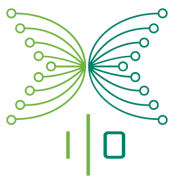
Description: All the features of the Business support plan, plus an assigned Technical Account Manager (TAM) who provides proactive guidance and best practices to help plan, develop, and run your AWS solutions, a Support Concierge who provides billing and account analysis and assistance, access to Infrastructure Event Management to support product launches, seasonal promotions/events, and migrations, and 15-minute response to critical support cases with prioritized case handling.

Price: Starts at \$15,000/month (scales based on usage)

If you select this option, customer support will contact you within 48 hours to discuss your needs and finalize the signup. Support resources will be available when signup is finalized, and no charges will be incurred until that time.

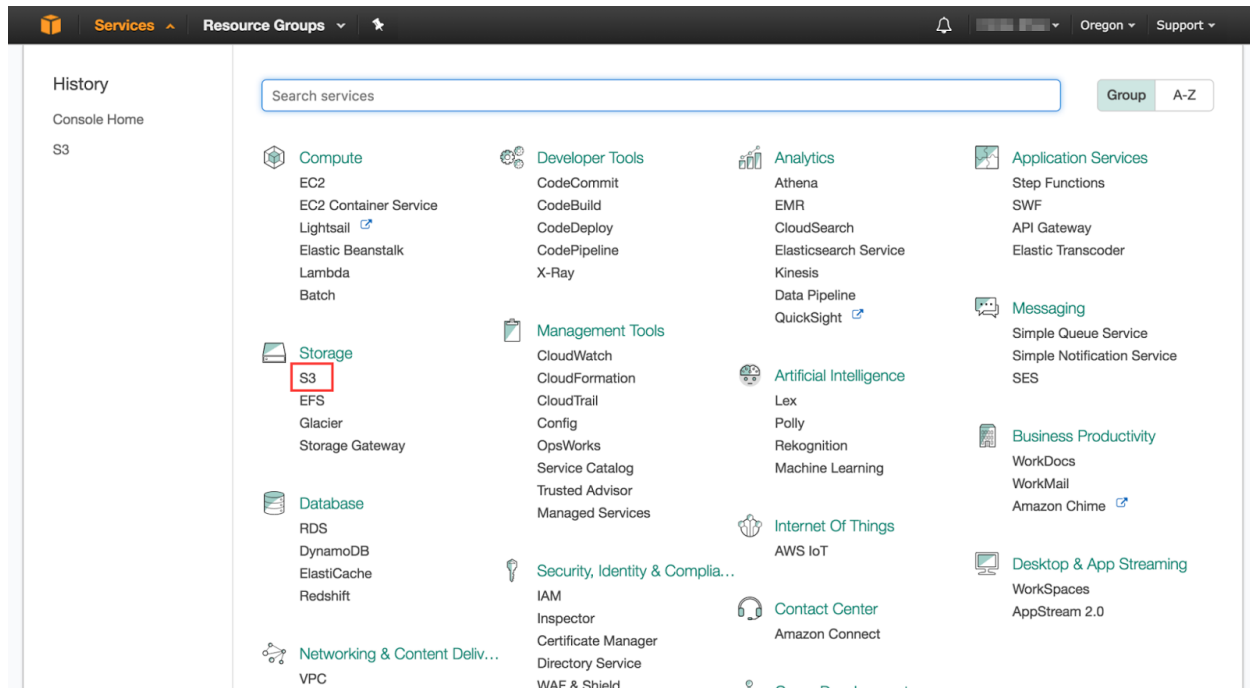
To explore all features and benefits of AWS Support, including plan comparisons and pricing samples, [click here](#).

Continue

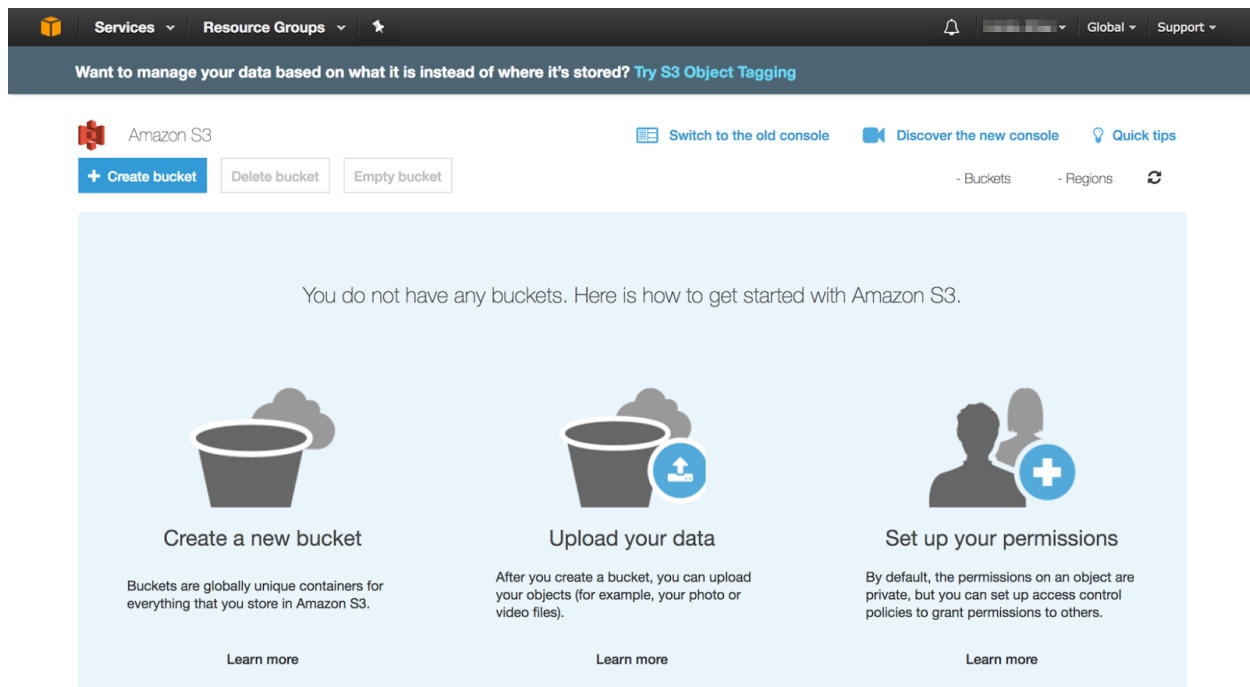


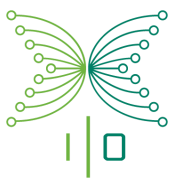
Sales App | Frontend - System setup

8. To enable S3, click “Services” and select S3.



9. Create a bucket by clicking “Create bucket”





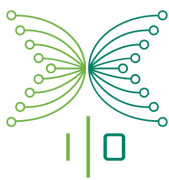
10. Input bucket name and click “Create”

The screenshot shows the AWS Management Console interface for creating a new S3 bucket. A modal window titled 'Create bucket' is open, displaying a four-step progress bar: 1. Name and region, 2. Set properties, 3. Set permissions, and 4. Review. The first step is active. It contains a 'Bucket name' input field, a 'Region' dropdown menu currently showing 'Asia Pacific (Singapore)', and a 'Copy settings from an existing bucket' dropdown menu showing 'You have no buckets'. At the bottom of the modal, there are 'Create', 'Cancel', and 'Next' buttons. The background shows the AWS console navigation bar and the Amazon S3 service page.

In order to use S3 buckets for the application, AWS credentials are needed.

11. On console menu, click your name on the top right and click “My Security Credentials”





12. Under Access Keys, click “Create New Access Key” to create an access key.

Your Security Credentials

Use this page to manage the credentials for your AWS account. To manage credentials for AWS Identity and Access Management (IAM) users, use the [IAM Console](#).

To learn more about the types of AWS credentials and how they're used, see [AWS Security Credentials](#) in AWS General Reference.

+

Password

+

Multi-Factor Authentication (MFA)

-

Access Keys (Access Key ID and Secret Access Key)

You use access keys to sign programmatic requests to AWS services. To learn how to sign requests using your access keys, see the [signing documentation](#). For your protection, store your access keys securely and do not share them. In addition, AWS recommends that you rotate your access keys every 90 days.

Note: You can have a maximum of two access keys (active or inactive) at a time.

Created	Deleted	Access Key ID	Last Used	Last Used Region	Last Used Service	Status	Actions
---------	---------	---------------	-----------	------------------	-------------------	--------	---------

Create New Access Key

Important Change - Managing Your AWS Secret Access Keys
As described in a [previous announcement](#), you cannot retrieve the existing secret access keys for your AWS root account, though you can still create a new root access key at any time. As a [best practice](#), we recommend [creating an IAM user](#) that has access keys rather than relying on root access keys.

+

CloudFront Key Pairs

+

X.509 Certificates

+

Account Identifiers

13. Please copy or download the keys, you will have to place these in the application's setting files.

Create Access Key

Your access key (access key ID and secret access key) has been created successfully.

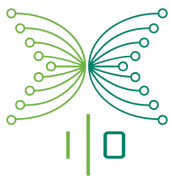
Download your key file now, which contains your new access key ID and secret access key. If you do not download the key file now, you will not be able to retrieve your secret access key again.

To help protect your security, store your secret access key securely and do not share it.

[Show Access Key](#)

Download Key File

Close



S3 configuration in file: `"/config/develop.json"`

```
"SLINGSHOT_S3_BUCKET": "*****",
"SLINGSHOT_MAX_FILE_SIZE_MB": 10,
"SLINGSHOT_S3_REGION": "ap-northeast-1",

"AWSAccessKeyId": "*****",
"AWSSecretAccessKey": "*****",
```

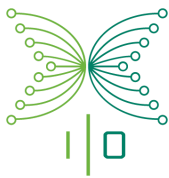
Loggly

For enabling logs on the server actions, it's required to add in Loggly configurations in secrets file, located at `"/config/${mode}/secret.json"`. "Enabled" field should be set to true and you should add in all required authentication credentials.

Loggly configuration in file: `"/config/test/secret.json"`

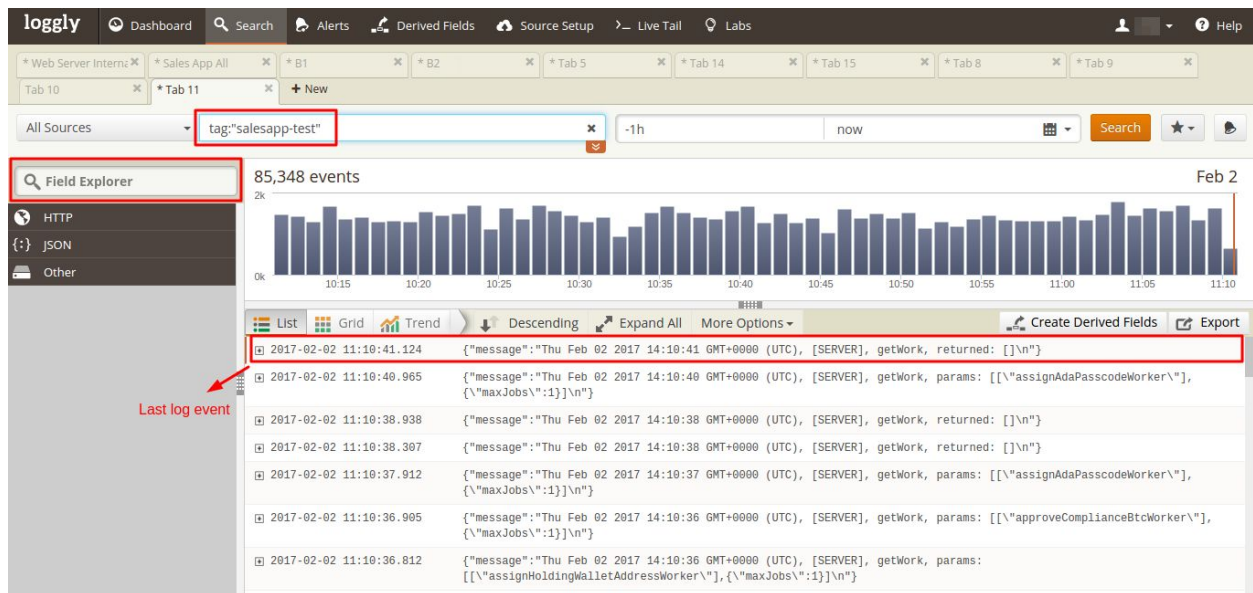
```
{
  "loggly": {
    "enabled" : true,
    "token": "*****_****_****_****_*****",
    "subdomain": "*****.loggly.com",
    "tags" : ["salesapp-test"],
    "auth": {
      "username": "*****",
      "password": "*****"
    }
  }
}
```

After the deploy is finished, if Loggly configuration was successful, you should be able to access the web page (in this example the url would be https://*****.loggly.com) and check that the logs are working.



Sales App | Frontend - System setup

Loggly search section:

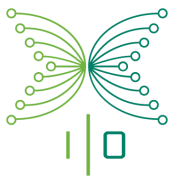


Mailgun

In order to allow Sales App to send emails using Mailgun service, it's needed to set the `MAILGUN_DOMAIN`, and `MAILGUN_API_KEY` as environment variables for Galaxy. They can be set in `/bin/${mode}/secret.json` file (the "mode" can be "production" or "test" depending on which environment will it be deployed).

Mailgun configuration in file: `/config/test/secret.json`

```
{
  "galaxy.meteor.com": {
    "env": {
      "MAILGUN_DOMAIN": "*****",
      "MAILGUN_API_KEY": "key-*****"
    }
  }
}
```



Recommendations

These are recommendations and experiences gathered by running the application through the presale of ADA.

Example Setup

This section will give an examples of setup, the setup given is the setup we are using in the development environment. The example setup will contain information of version of configuration settings that have been mentioned in this document. Example settings files and deployment scripts will also be provided.

Hosting Services Configuration

These are the settings we use for different hosting services.

Galaxy

Scale up to 3 containers, enforce https, and use a custom certificate.

Compose

Enable ssl and oplog, and use MMAPv1 as storage engine.

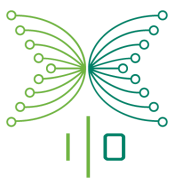
Slack

Create a custom slack channel webhook for receiving deployment notifications.

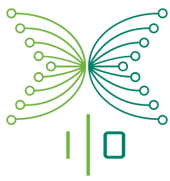
Settings

settings.json

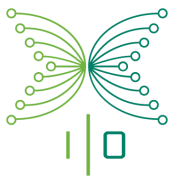
```
{
  "public": {
    "salesOver": false,
    "development": false,
    "salesStarted": true,
    "expireTickets": {
      "bank": {
        "afterInvoiceSent": {
          "days": 3,
          "businessDays": true
        }
      }
    },
    "btc": {
```



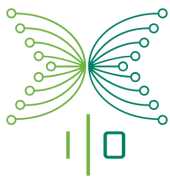
```
        "afterInvoiceSent": {
          "days": 3,
          "businessDays": false
        }
      },
    },
    "acceptInviteBtcTimer": 1800,
    "minUsdOrderValue": 1000,
    "persistent_session": {
      "default_method": "persistent"
    },
    "salesLimits": {
      "currentTranche": "t4",
      "tranches": {
        "t1": {},
        "t2": {},
        "t3": {},
        "t3.5": { "totalAmountAvailable": 8000000, "overCapacityTolerance": 0.25 },
        "t4": { "totalAmountAvailable": 30000000, "overCapacityTolerance": 0.20 }
      }
    },
    "availableCountries": ["JP", "KH", "CN", "KR", "PH", "TH", "VN", "MY", "TW"],
    "availableLanguages": ["en", "ja", "ko", "zh"],
    "residenceCountriesAllowedSale": ["JP", "KH", "CN", "KR", "PH", "TH", "VN", "MY", "TW"],
    "support": {
      "ja": {
        "email": "support@*****"
      },
      "ko": {
        "email": "korean@*****"
      },
      "zh": {
        "email": "chinese@*****"
      },
      "default": {
        "email": "support@*****"
      }
    },
    "regionalMappings": {
      "JP": "regionNippon",
      "KR": "regionSouthKorea",
    },
    "regionalSettings": {
      "regionNippon": {
        "enrollment": {
          "paymentOptions": ["bank", "btc"],
          "paymentMinimum": 1000
        },
      },
    },
  },
}
```

```
"latestPolicyUpdate": {
  "buyer": {
    "TOC": "2016-11-26"
  },
  "distributor": {
    "0": "2016-02-04",
    "1": "2016-02-04",
    "2": "2016-02-04",
    "3": "2016-02-04"
  },
  "PRIVACY": "2015-10-01"
},
"policies": {
  "TOC": {
    "distributor": {
      "0": [{ "lang": "ja", "file": "appoiu.pdf" }],
      "1": [{ "lang": "ja", "file": "t1k4j3kkjh4l3k4.pdf" }],
      "2": [{ "lang": "ja", "file": "t2j3j9dadflk34.pdf" }],
      "3": [{ "lang": "ja", "file": "t3y4y2y5uhkjh.pdf" }]
    },
    "buyer": [{
      "lang": "ja",
      "file": "toCforbuyerJP_v5a_jp_userpoli.pdf"
    }],
    "default": [{
      "lang": "ja",
      "file": "toCforbuyerJP_v5a_jp_userpoli.pdf"
    }]
  },
  "RISK": {
    "default": [{
      "lang": "ja",
      "file": "riskPolicyJP_v2a_jp_rskpoli.pdf"
    }]
  },
  "PRIVACY": {
    "default": [{
      "lang": "ja",
      "file": "privacy.pdf"
    }]
  }
},
"regionSouthKorea": {
  "latestPolicyUpdate": {
    "buyer": {
      "TOC": "2016-11-26"
    },
```

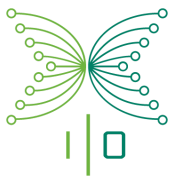


```
"distributor": {
  "0": "2016-09-20",
  "1": "2016-09-20",
  "2": "2016-09-20",
  "3": "2016-09-20"
},
"PRIVACY": "2016-09-20"
},
"enrollment":{
  "paymentOptions": ["btc"],
  "paymentMinimum": 1000
},
"policies": {
  "TOC": {
    "distributor": {
      "0": [{
        "lang": "en",
        "file": "kr-toc-dist-t0.pdf"
      }],
      "1": [{
        "lang": "en",
        "file": "kr-toc-dist-t1.pdf"
      }],
      "2": [{
        "lang": "en",
        "file": "kr-toc-dist-t2.pdf"
      }],
      "3": [{
        "lang": "en",
        "file": "kr-toc-dist-t3.pdf"
      }]
    },
    "buyer": [{
      "lang": "en",
      "file": "toCforbuyerEN_v5a_en_userpoli.pdf"
    }],
    "default": [{
      "lang": "en",
      "file": "toCforbuyerEN_v5a_en_userpoli.pdf"
    }]
  },
  "RISK": {
    "default": [{
      "lang": "en",
      "file": "riskpolicyEN_v2a_en_rskpoli.pdf"
    }]
  },
  "PRIVACY": {
```



```
        "default": [{
          "lang": "ko",
          "file": "privacy-kr.pdf"
        }]
      }
    },
    "regionOther": {
      "enrollment": { "paymentOptions": ["btc"], "paymentMinimum": 1000 }
    }
  },
  "dollarsPerAda": 0.0026,
  "disableJobCollectionLogger" : false,

  "btcPriceCriteria": {
    "providersRanking": [
      "bitstamp",
      "blockchain",
      "bitpay",
      "bitgo",
      "bitfinex",
      "kraken"
    ],
    "outdatedMaxToleranceInMinutes": 2
  }
}
```



secret.json

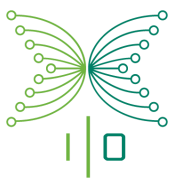
```
{
  "SLINGSHOT_S3_BUCKET": "*****",
  "SLINGSHOT_MAX_FILE_SIZE_MB": 10,
  "SLINGSHOT_S3_REGION": "ap-northeast-1",

  "AWSAccessKeyId": "*****",
  "AWSSecretAccessKey": "*****",
  "SALES_APP_AUTH_API_TOKEN": "*****_*****_*****_*****",

  "loggly": {
    "enabled" : true,
    "token": "*****_****_****_****_*****",
    "subdomain": "*****.loggly.com",
    "tags" : ["salesapp-test"],
    "auth": {
      "username": "*****",
      "password": "*****"
    }
  },

  "kadira": {
    "appId": "*****",
    "appSecret": "*****_****_****_****_*****"
  },

  "galaxy.meteor.com": {
    "env": {
      "ROOT_URL": "http://*****",
      "MONGO_URL":
"mongodb://*****@*****,*****
*****/*****?ssl=true&authMechanism=SCRAM-SHA-1",
      "MONGO_OPLOG_URL":
"mongodb://oploguser:*****@*****
*****/local?authSource=admin&ssl=true",
      "INVOICE_FOLDER": "../invoices/",
      "MAILGUN_DOMAIN": "*****",
      "MAILGUN_API_KEY": "key-*****",
      "IOHK_BACKEND_SERVER_IP": "https://*****/api/v1/",
      "ATTAIN_BCC_EMAIL": "test-email+test-attain@iohk.io",
      "IOHK_BACKEND_CLIENT_ID": "SalesAppTesting",
      "IOHK_BACKEND_TOKEN": "*****_****_****_****_*****",
      "IOHK_BACKEND_SIGNATURE": "*****_****_****_****_*****"
    }
  }
}
```



Scripts

To ease up deployment to testing and development environment we used following scripts. They can be modified to suite any environment similar to our. The scripts are located at `./bin/deploy` folder, and they are wrapping the more general deploy script.

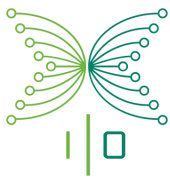
<i>to-production-config</i>	Creates a configuration file for production. The file will be save in the temp folder and the folder will be printed in the terminal.
<i>to-production-deploy</i>	Deploy to production using configuration file, which needs to be passed in as a parameter. You can use <i>to-production-config</i> to generate this config file.
<i>to-production-no-config</i>	Deploy to production without updating the configuration file.
<i>to-test-attain-iohk-io</i>	Deploys to test.attain.iohk.io server with the default settings file. Since this script is setting test mode, the minimum purchase is overwritten to become 1 USD.
<i>to-test-iohk-io</i>	Deploys to test.iohk.io server with the default settings file. Since this script is setting test mode, the minimum purchase is overwritten to become 1 USD.

Known Quick Fixes

The application is not always running smoothly when scaled to run multiple instances in parallel against the same database. If it's still desireable to do so, then these quick fixes have proven to solve the situation regardless of what the initial problem was for a number of times. However, there is no guarantee they will work, and in some cases they might cause more issues than they solve.

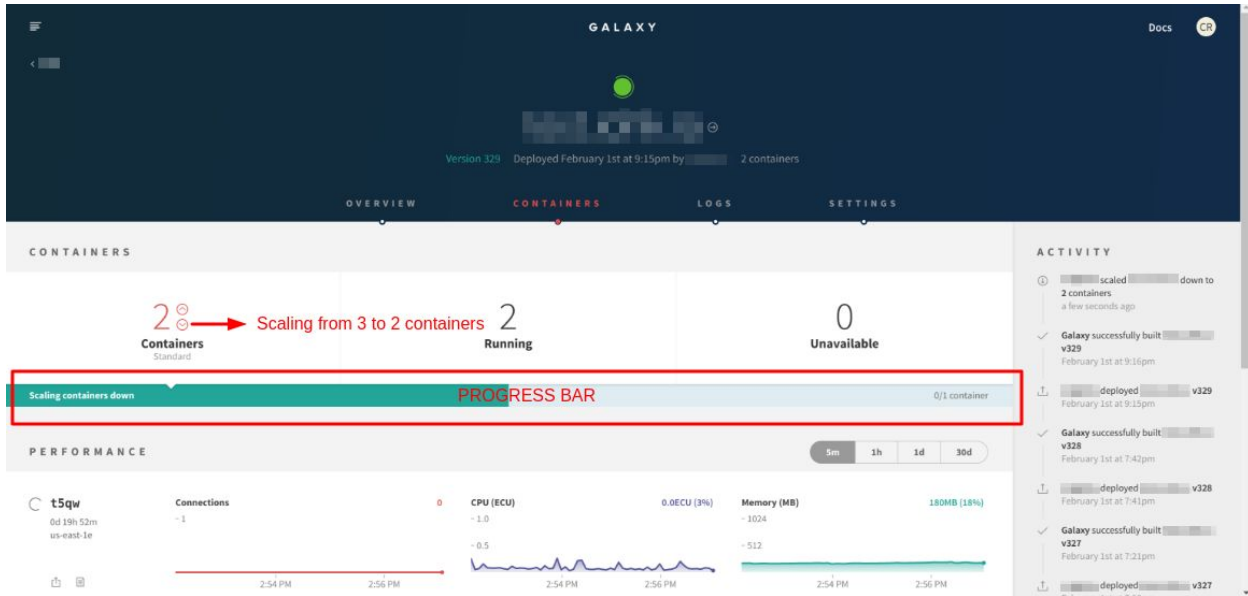
Scaling Down and Up Container

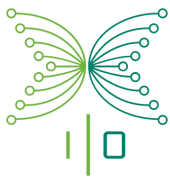
This fix assumes use of Galaxy as cloud host. By simply scaling down the container size by one container, and then scaling up again some issues seems to disappear magically. This is really useful when a worker has stalled (see System operation in Frontend - Business logic).



Scale Down

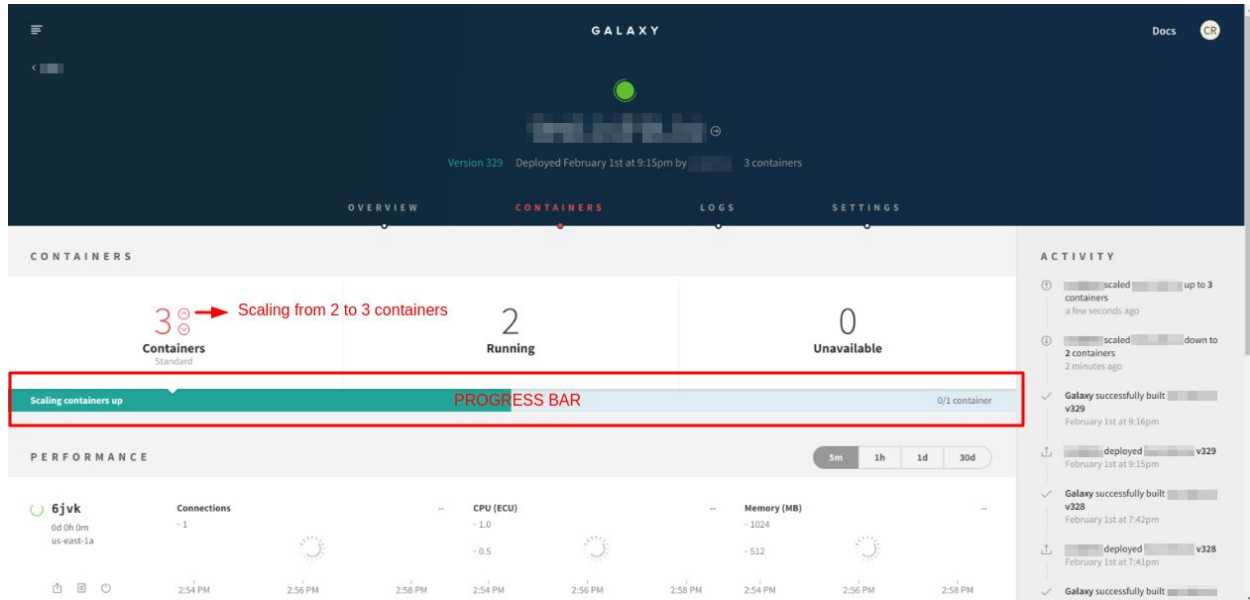
Press the down arrow button to scale down the amount of containers by one.

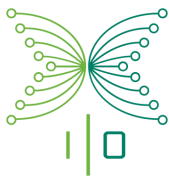




Scale Up

Press the up arrow button to scale up the amount of containers by one.





Restart the Application

This quickfix has so far fixed any issue not solved by scaling up and down containers (see section above), but with the drawback that the application will go offline for a couple of minutes. Simply shut down all running instances and start them up again one by one. The steps below is for doing it in Galaxy.

Stop the containers

Navigate to the bottom of the settings page.

you are currently using 3 standard containers. Scale your app by changing the size of containers now.

DOMAINS & ENCRYPTION

Apps deployed to Galaxy get a unique Galaxy Domain. Add custom domains to make your app accessible via other domain names then configure your DNS provider to point at Galaxy's DNS us-east-1.galaxy-ingress.meteor.com. We recommend that you also enable encryption for your domains to secure sensitive data.

Domains	Encryption
<div><div></div><div></div><div></div></div> <div>+ Add new domain</div>	<div>Enabled</div> <div>Primary Domain</div>

DANGER ZONE

Stop app

Stopping an app shuts down all of its containers. When stopped, the app's settings are preserved and organization members may continue to view app logs and performance metrics.

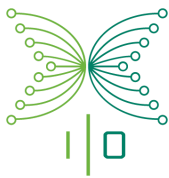
Click here

STOP

Delete app

Deleting an app shuts down all containers and removes the app from Galaxy.

DELETE



Sales App | Frontend - System setup

Click “stop” followed by “confirm stopping”.

Apps deployed to Galaxy get a unique Galaxy Domain. Add custom domains to make your app accessible via other domain names then configure your DNS provider to point at Galaxy's DNS us-east-1.galaxy-ingress.meteor.com. We recommend that you also enable encryption for your domains to secure sensitive data.

Domains

Encryption

Enabled

Primary Domain

+ Add new domain

DANGER ZONE

Stop app

Stopping an app shuts down all of its containers. When stopped, the app's settings are preserved and organization members may continue to view app logs and performance metrics.

Are you sure you'd like to stop test.iohk.io?

CANCEL

STOP APP

Confirm stop

Delete app

Deleting an app shuts down all containers and removes the app from Galaxy.

DELETE

Navigate to the containers tab

6jvk

0d 0h 25m

us-east-1a

Connections

0

CPU (ECU)

0.1ECU (8%)

Memory (MB)

123MB (12%)

3 Containers

0 Running

0 Unavailable

Scaling containers down

Stopping status

0/3 containers

ACTIVITY

stopped

a minute ago

scaled up to 3 containers

26 minutes ago

scaled down to 2 containers

27 minutes ago

Galaxy successfully built v329

February 1st at 9:16pm

deployed v329

February 1st at 9:15pm

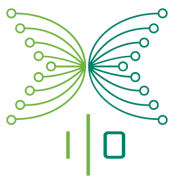
Galaxy successfully built v328

February 1st at 7:42pm

deployed v328

February 1st at 7:42pm

Wait for the containers to stop



Sales App | Frontend - System setup

Containers are fully stopped!

Version 329 Deployed February 1st at 9:15pm by [redacted] 3 containers

OVERVIEW CONTAINERS LOGS SETTINGS

CONTAINERS

3 Containers Standard

0 Running

Empty status

0 Unavailable

PERFORMANCE

Showing all containers

5m 1h 1d 30d

ACTIVITY

- stopped 2 minutes ago
- scaled up to 3 containers 27 minutes ago
- scaled down to 2 containers 28 minutes ago
- Galaxy successfully built v329 February 1st at 9:16pm
- deployed v329 February 1st at 9:15pm
- Galaxy successfully built v328 February 1st at 7:42pm
- deployed v328 February 1st at 7:41pm

Start the containers

Navigate to the bottom of the settings page and click “start”.

Container size: standard

You are currently using 3 Standard containers. Scale your app by changing the size of containers now.

CHANGE

DOMAINS & ENCRYPTION

Apps deployed to Galaxy get a unique Galaxy Domain. Add custom domains to make your app accessible via other domain names then configure your DNS provider to point at Galaxy's DNS us-east-1.galaxy-ingress.meteor.com. We recommend that you also enable encryption for your domains to secure sensitive data.

Domains Encryption

Enabled Primary Domain

+ Add new domain

DANGER ZONE

Start app

Start [redacted] with 3 containers

Click here

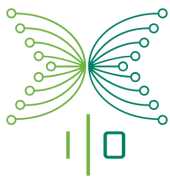
START

Delete app

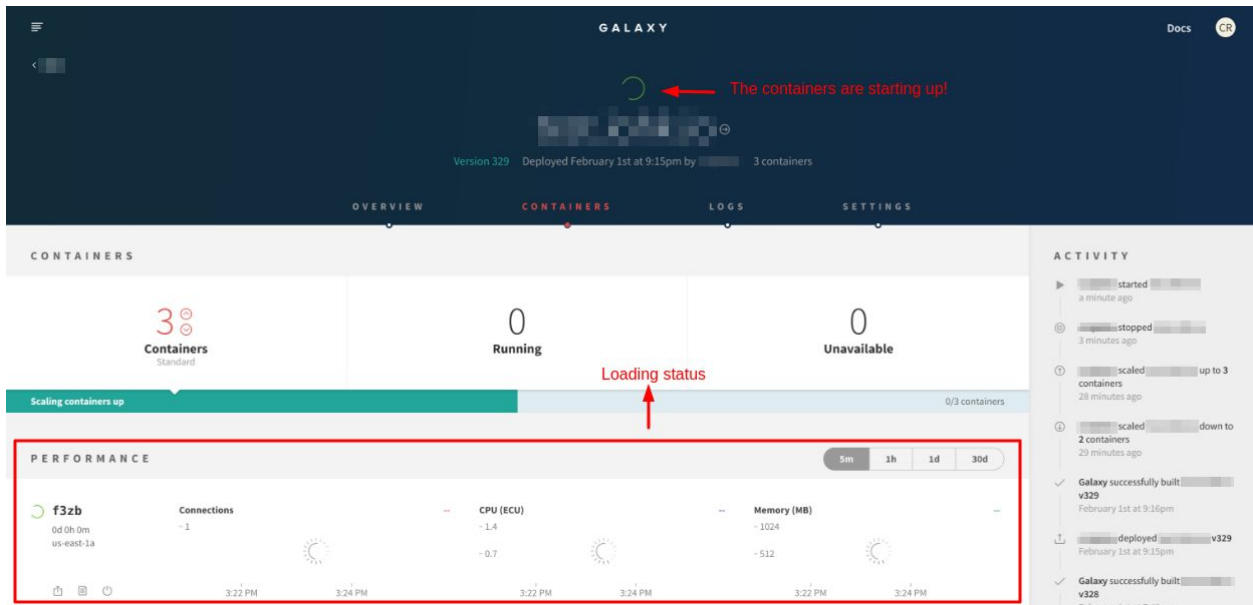
Deleting an app shuts down all containers and removes the app from [redacted] Galaxy.

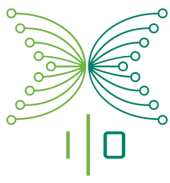
DELETE

Navigate to the containers tab and wait for the containers to start up again.



Sales App | Frontend - System setup





Sales App | Frontend - System setup