IT Architecture

RampedUp is deployed using Amazon Simple Storage Service (Amazon S3). All of the assets of this AngularJS application are deployed to, and served from, an S3 bucket. The deployed web application interacts with the application's back-end services through RESTful calls that are routed through Amazon API Gateway, supplying tenant identity context with each call.

The technology stack is ASP.Net, C# as the main language, with an AngularJS client framework. The database is MySQL and hosted on AWS:

- Elasticsearch Service
- Key Management Service
- Relational Database Service
- Simple Email Service
- Simple Storage Service (3)
- Elastic Compute Cloud (2)

The core of the SaaS application's services are hosted in the VPC's private subnets. An Amazon ECS cluster hosts the containers that run the system's microservices. Seven separate Node.js microservices are deployed in this cluster. This cluster also employs Auto Scaling for basic high availability. The cluster is tuned to dynamically respond to changes in tenant load, scaling up and down based on demand. Each service applies the context of a tenant's identity to control and scope access to the system's resources.

The reference application uses a variety of AWS services; for example:

- Amazon DynamoDB tables are provisioned in a multi-tenant model for services that require storage.
- AWS Identity and Access Management (IAM) manages and applies isolation polices and roles to prevent cross-tenant access.
- Amazon Cognito serves as the identity provider, storing attributes that identify each tenant.
- Amazon Simple Notification Service (Amazon SNS) publishes validation emails during the user registration process.

The battlecard core is a .Net C# application fully hosted in AWS. The web application uses the Asp.Net Web API framework. The front end is a Single Page Application written in AngularJS which communicates to the Web API controllers via http. The service layer where all of the business logic and database access occurs is accessed from the Web API controllers via dependency injection. The application database is Amazon Aurora and is made accessible by our application using Entity Framework. Full text search functionality is supported by Elasticsearch, which is hosted in AWS Elasticsearch Service. Queued and scheduled background jobs are run in custom C# Windows services accessing the same service layer.

Intrusion Detection & Prevention Systems

RampedUp security is designed into multiple layers of the AWS RampedUp environment with a data center and network architecture. AWS takes responsibility for security of the cloud while RampedUp, on the other hand, is responsible for security configuration base on multi-tenancy and usage norms.

Intrusion Detection Systems (IDS) monitor systems for malicious activity and report them to RampedUp administrators. Intrusion Prevention Systems (IPS) are positioned behind firewalls and provide an additional layer of security by scanning and analyzing suspicious content for potential threats. Placed in the direct communication path, an IPS will take automatic action on suspicious traffic within the network.

Safeguards:

We have decoupled our app from the website as stated for best practices DDoS. Our system has a load balancer and elastic capacity should a spike occur. We are notified by AWS when there is an excessive spike to isolate and mitigate the attack.

We limit requests to business email addresses to access our system – limiting personal email addresses as a result. This process then notifies a RampedUp employee to look up every request to send an activation link – the process is not automated. We created this process as a direct response to the lack of control by automating the process.

End Point Security

- All development work is done in virtual environment.
- Workstations auto-lock after 60 seconds of dormant use.
- Device passwords require 2 Factor Authentication for recovery. Passwords must be reset every 90 days, cannot be a previously used password, and require 8 digits.
- Workstations and laptops must adhere to virus and malware protection policy – McAfee
- Laptops are wiped clean after employment is terminated.

Change and Control Process

1. Define the Change Request

A Change Request is the documentation used to request the actual change. Whoever owns the actual request needs to explain it in such a way that the team understands it well enough to define it. This should be done through appropriate documentation (whatever the project team or company expects). It can be as simple as an email or as complex as a formal document.

When defining the change, it's necessary to have in hand the actual request with all supporting statements. This should include:

- **Actual Request**: Statement of the need. This should outline clearly the change item for the project team to analyze.
- **Reason for the Request:** Customer impacts if the request cannot be completed or if considerable time passes before the request can be completed
- **Conditions of Success**: While this is an Agile term, I believe it's valid with Waterfall, as well. Call it what you want, but customers must be able to define what they expect from the change.
- **Expected Completion:** The requester should provide an expected due date for the item. This doesn't mean the change will be completed by this date. It's simply meant to provide more details for the team to analyze when defining options.

- **Expected Value:** This should explain why the request is needed. It can either be something as simple as “better customer experience” or “revised calculation provides more accurate data” in relation to a report.

2. Submit and Review the Change Request.

Once the Change Request is documented, it’s submitted to the project team. Here again, the process varies from the simple (a phone call or email) to the formal (a memo or meeting). Unless the request is very simple, I prefer to review the change with the full team. That meeting provides the proper venue for the request to be reviewed, and all members have a chance to ask questions and help make decisions.

There should be two portions to reviewing the Change Request: the formal presentation or meeting and the project team’s review and discussion of impacts. Within the change control process there should be an expected turnaround time for these. Discussions with the customer should include setting expectations regarding response time, or at least when the team will provide feedback.

3. Define Options and Create Response Document

Once the team has reviewed the Change Request, options should be defined. There should be a minimum of two. When providing the document response, always provide each option with some of the data points below as well as a team recommendation, which represents its view of the best choice. The customer may not always go along, but it can help them make a decision.

The response should include:

- Option Number and Name
- Proposed Solution: This should include how to respond to the change request. It can be anything from a technical direction and justification as to why this particular approach is being put forward.
- Proposed Timeline: The customer always needs to know how long something is going to take. The estimated timeline is a piece of information they will leverage when making a choice based on the options the team presents.
- Impacts to the Project: This is an essential part of the response. If changes are small, there may be no impacts — for instance, if you’re changing a series of messages or buttons. But most changes will have some sort of impact. The scope change can impact the timeline, the budget and therefore the quality of the product. This area should minimally explain the cost of the changes, the impact on the timeline and potential quality results. There may also be resource impacts. The team may either have to get additional people or may define a need for existing resources to add or remove time on the project. All of these items should be defined clearly to enable the customer’s decision making.
- Expiration Date for Proposed Changes: This sets a timeframe for the client to respond to the proposed solution and cost/time impacts. If the client goes outside of the set window, there could be additional impacts to the project. That aside, setting an expiration date provides urgency to the process.

4. Final Decision And Approval

The customer should provide a timely response. If the Change Control Response document expires, it should be re-evaluated once the customer provides feedback. If too much development has occurred to sustain the change, then that needs to be stated. If the delayed response has resulted in other impacts,

they need to be communicated as soon as possible. It's also possible that an expired response could lead to an additional review and proposal.

Whatever decision results from all this needs to be officially approved. When you define the Change Control process, be sure to include a list of sponsors, stakeholders and key decision makers who can OK both the process and the decision.

Salesforce SSO policy - https://help.salesforce.com/articleView?id=sso_about.htm&type=5

> Password Policy: RampedUp offers Single Sign On with Salesforce and the ability to create a User Name and Password. If choosing Salesforce then the parameters are consistent with that platform. If choosing RampedUp the parameters are as follows. An account must be created by RampedUp admin and emailed to user. The password is autogenerated and randomized. The user can change their password within the app it is at least 6 characters. The user can request a new password emailed to them that will again be autogenerated and randomized. RampedUp employees do not have access to nor can they change passwords.