**Disaster Recovery Plan**

**GitHub Enterprise Services**

# Background and Context:

The purpose of this document is to elaborate the process and procedures that should be in place for ensuring that GitHub is always available within mmm.com domain.

# Objectives:

* To minimize interruptions to the normal operations using GitHub.
* To limit the extent of disruption and damage of the projects and code repositories in GitHub.
* To minimize the economic impact of the interruption.
* To establish alternative means of operation in advance.
* To train personnel with emergency procedures.
* To provide for smooth and rapid restoration of service.

# Personnel

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Position** | **Role** | **Email / Contact number** |
| Paul Pottorff |  | Approver | [ppottorff@mmm.com](mailto:ppottorff@mmm.com) |
| Troy Gotch |  | Reviewer | [tgotch@mmm.com](mailto:tgotch@mmm.com) |
| Pratyush Mishra |  | Scrum Master | pmishra4.cw@mmm.com |
| Niket Chandrawanshi |  | Co-Owner of Backup / Restore Procedures | Nchandrawanshi.cw@mmm.com |
| Roopa Gowda |  | Co-Owner of Backup / Restore Procedures | rmgowda.cw@mmm.com |

# GitHub Organization – Regions in Scope

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Region** | **Environment** | **Location** | **Primary** | **Backup** |
| **East-US-2-A** | Production | Ohio | Yes | N/A |
| **East-US-2-B** | Production Backup | Ohio | N/A | Yes |
| **East-US-1-A** | Stage | N Virginia | Yes | N/A |
| **East-US-1-B** | Stage Backup | N Virginia | N/A | Yes |

# Environments in Scope:

1. PROD
2. STAGE

Note: Both these environments are available in all regions listed above.

Initial Planning

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Region** | **Environment** | **Backup Frequency** | **Backup Retention Policy** | **Audit Frequency** | **Notifications** |
| **East-US-2** | Production | Hourly (Incremental) | 30 days | Yearly | Backup of the primary server is taken in an hourly incremental mode for PROD. |
| **East-US-1-A** | Stage | Once a day (Incremental) | 30 days | Yearly | Backup of the primary server is taken in an hourly incremental mode for STAGE. |

# Failure Modes / Scenarios

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Events** | **Recovery Plan** | **Recovery Time Objective (RTO)** | **Recovery Point Objective (RPO)** | **Action Taken** |
| **The primary server is down** | 1. [Elevate a replica node to the primary role](https://docs.github.com/en/enterprise-server@3.8/admin/enterprise-management/configuring-high-availability/initiating-a-failover-to-your-replica-appliance) and [demote the old primary.](https://docs.github.com/en/enterprise-server@3.8/admin/enterprise-management/configuring-high-availability/recovering-a-high-availability-configuration) 2. Ensure functionality of the application. 3. Troubleshoot all issues with the primary server to restore service. Open a ticket with GitHub vendor support if needed. 4. If the old primary server cannot recover, [create a new replica server](https://docs.github.com/en/enterprise-server@3.8/admin/enterprise-management/configuring-high-availability/creating-a-high-availability-replica) (being sure to start replication). 5. Notify end-users of the impact via Microsoft Teams and email. 6. Identify the root cause of the issue and update corrective / preventive measures to prevent the issue from happening in the future. | 30Min | 30min – 1hr | We have replica servers in EAST-US-2-B**,** once primary region gets down traffic will be redirect to EAST-US-2-B region. |
| **A replica server is down** | 1. Troubleshoot all issues with the replica server to restore service. Open a ticket with GitHub vendor support if needed. | 24-48 hours | 0Hr |  |
| **The backup EFS cluster is down** | 1. Troubleshoot all issues with the backup cluster to restore service. Open a ticket with GitHub vendor support if needed. 2. If we can’t recover the backup cluster, we recreate the cluster using following script <SCRPIT URL -- CF > 3. Perform a root cause analysis (RCA) and Impact analysis. 4. Identify the root cause of the issue and update corrective / preventive measures to prevent the issue from happening in the future. | 24-48Hrs | 0Hr | We can recreate the EFS share and sync the files from s3 bucket, if required. **Note**: Our cluster are saved data in EFS and get synced with S3, in every 2hr using data sync service. |
| **Deactivated GitHub Account (individual)** | 1. Contact GitHub support via ticket and phone to re-instate GitHub Account. 2. Contact end-users with cause of outage and estimated recovery time. 3. Perform Root cause analysis (RCA) and Impact analysis. 4. Update Corrective / Preventive measures in the recovery plan. | 12-24 hours | 12-24 hours | Connect to GitHub Support team for recovery |
| **Code Blue Scenarios** | | | |  |
| Failed Upgrade | Roll back of the update occurs automatically. If the update package needs to be re-download, please follow the below steps  Download the upgrade package to your GitHub Enterprise Server instance using curl:  - “curl -L -O UPGRADE-PKG-URL”  Run below command to recover the previous package:  - “ghe-upgrade --allow-patch-rollback EARLIER-RELEASE-UPGRADE-PACKAGE.pkg” | 4-6 hours | 0 hour |  |
| AWS Outage | 1. Notify users of AWS outage, updates from AWS, and estimated recovery time. 2. Raise High priority ticket with AWS and follow the steps suggested by AWS support team. 3. Below teams have admin access to console to perform on support actions   - AWS-GITHUB02-Administrators  - AWS-GITHUB101-Administrators - AWS-GITHUB102-Administrators | 4-6 hours | 1 hour |  |
| Infrastructure failure | 1. Notify users of infrastructure failure on AWS, updates from AWS, and estimated recovery time. 2. Raise High priority ticket with AWS and follow the steps suggested by AWS support team. 3. If the underlying infrastructure will continue to be compromised beyond the RTO, recreate any effected infrastructure by re-running the [GitHub infrastructure CICD pipeline](https://github.com/3M-Cloud/DPCoE-GitHub-Migration-CT). 4. Ensure functionality of the application. 5. Perform Root cause analysis (RCA) and Impact analysis. | 10-12 hours | 1 hour |  |
| **AWS Account deleted** | 1. Restore account by approaching AWS support to restoring AWS account. 2. Perform Root cause analysis (RCA) and Impact analysis. 3. Update Corrective / Preventive measures in the recovery plan. | 24-72 hours | 0 hour | Raise the High Priority ticket with AWS Support  Then connect with platform engineering. team |
| **Loss of service in GitHub.com** | 1. Notify users of the outage, updates from GitHub, and estimated recovery time. 2. Check <https://www.githubstatus.com/> and contact GitHub support via ticket and phone if the issue hasn’t been noted by GitHub. | 24-72 hours | 1 hour | Raise Ticket with GitHub Support for recovery. |

# Disaster Action Checklist

* 1. Notify stakeholders
  2. Contact and set up disaster recovery team
  3. Determine degree of disaster
  4. Implement proper application recovery plan dependent on extent of disaster
  5. Monitor progress
  6. Contact all other necessary personnel
  7. Contact vendors–both hardware and software

# Dry Runs / Simulating the Failure Modes / Scenarios

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item** | **Yes** | **No** | **Applicable** | **Not Applicable** |
| *Conducting a Recovery Test* |  |  |  |  |
| 1. In case of disk utilization threshold crossed? 2. In case of CPU utilization threshold crossed? 3. Backup server disk utilization threshold crossed? 4. Backup issues? 5. DNS/Network related issue | Yes  Yes  Yes  Yes | No | 1. New relic Alerts are already setup. 2. New relic Alerts are already setup. 3. New relic Alerts are already setup. 4. Setup AWS S3 sync every 2 hrs and have a yearly check on backups. 5. Review and troubleshoot own infrastructure and if required we’ll raise ticket with DNS team. |  |
| *Areas to be tested* |  |  |  |  |
| 1. In case of primary instance down, traffic is redirected into another node. 2. Backup effectiveness on prod instances 3. How many request our instance can handle? 4. In case of DDOS Attack? 5. Ability to recover and process successfully without key people. 6. Ability of the plan to clarify areas of responsibility and the chain of command. 7. Effectiveness of security measures and security bypass procedures during the recovery period. 8. Ability of users to continue day-to-day operations without applications or jobs that are considered noncritical. 9. Ability to contact the key people or their designated alternates quickly. 10. Ability to adapt plan to lesser disasters. | Yes  Yes  Yes  Yes  Yes  Yes  No  Yes  Yes  Yes |  | 3. 999999999  4. Relying on AWS infrastructure and to prevent the DDOS attacks  5. Documentation is in place and the user should be added to GitHub admin groups.  6. Documentation is in place  7. Following 3M security requirements there is no security bypass procedures  9. Documentation is in place  10.Yes, we can adapt it if needed. |  |

# Logging / Audit Trail

Maintain a log of all events and dry run tests to ensure that there’s a record of all changes made to the GitHub environment.

|  |  |  |
| --- | --- | --- |
| **Dry run test** | **Day performed** | **Result** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# GitHub Backup Process

From EC2 Backup Server

[GitHub Enterprise Backup and Restore - Digital Product Center of Excellence - 3M Wiki Enterprise (mmm.com)](https://we.mmm.com/wiki/display/CHBS/GitHub+Enterprise+Backup+and+Restore)

From ECS Clustered Backups   
[GitHub Restoration Documentation](https://we.mmm.com/wiki/display/CHBS/GitHub+Restoration+Documentation)