IPPC NextGen Project Documentation

Version:	Draft 0.9
Prepared by:	Various EPIC team members
Submitted on:	December 30, 2021
Reviewed on date:	_____
Reviewed by:	_____, IPPC (list names of IPPC team members) _____, IPPC (list names of IPPC team members) _____, IPPC (list names of IPPC team members)

Content Overview:

1. Project overview (short description)
2. Main components schemas and description
    1. **PC**: Client Device, Preprocessor, Legacy System, NextGen Database, NextGen Processing, NextGen Dashboard
    2. **Android/Mac:** Device client, Mobi database, NextGen Database, NextGen Processing, NextGen Dashboard
    3. **iOS**: iCloud, Reincubate, WebHook, NextGen Database, NextGen Processing, NextGen Dashboard
3. Quick overview for NextGen Dashboard User Interface
    1. Case
    2. Device
    3. Officer
    4. Event
    5. Swimlanes
    6. Reports
        1. Swimlane expand report
        2. Timeline/Context report
4. Case / Device / Officer
    1. What are the cases / devices / officers
    2. Types of Devices
    3. How we store cases/officers/devices in database
5. Events
    1. Type of events
    2. How we store events in database (for for different event types)
    3. How different event types look on UI
    4. References: tagid-sourceid mapping for events
6. PC NextGen Processing
    1. PC processing schema (components and flows for PC processing)
    2. How PC events are created in the database
    3. How we process different event types
        1. Web/WebSearch/YouTube
        2. Screenshots
        3. Blocked Web/App
        4. External Disks
        5. Keystrokes
    4. Code components & deployment notes
    5. AWS PC processing infrastructure (EB profiles, Redis, Database)
7. Android/Mac NextGen Processing
    1. Android/Mac processing schema (components and flows for PC processing)
    2. How Android/Mac event created in NextGen database
    3. Mobi database
    4. How we process different event types
        1. Web/WebSearch/YouTube
        2. Screenshots/Photos/Videos
        3. Blocked Web
        4. SMS/MMS
        5. Calls
        6. Keystrokes

5. Code components & deployment notes
6. AWS PC processing infrastructure (EB profiles, Redis, Database)

8. iOS NextGen Processing
    1. iOS processing schema (components and flows for PC processing)
    2. How iOS event created in database
    3. How we process event different event types
        1. Photos/Videos
    4. Code components & deployment notes
    5. AWS PC processing infrastructure (EB profiles, Redis, Database)
9. Specific processing
10. DevOps Notes
    1. Elastic Beanstalk (EB)
    2. How to create a new EB environment
    3. How to deploy a new app to EB environment
    4. How to deploy an existing app to EB environment
    5. How to scale up/down EB environment
    6. How stop an EB environment
    7. How we using cloud watch for logging
    8. Reprocessing
    9. Monitoring dashboard

Notes:

Table of Contents

IPPC Technical Documentation

Project Overview

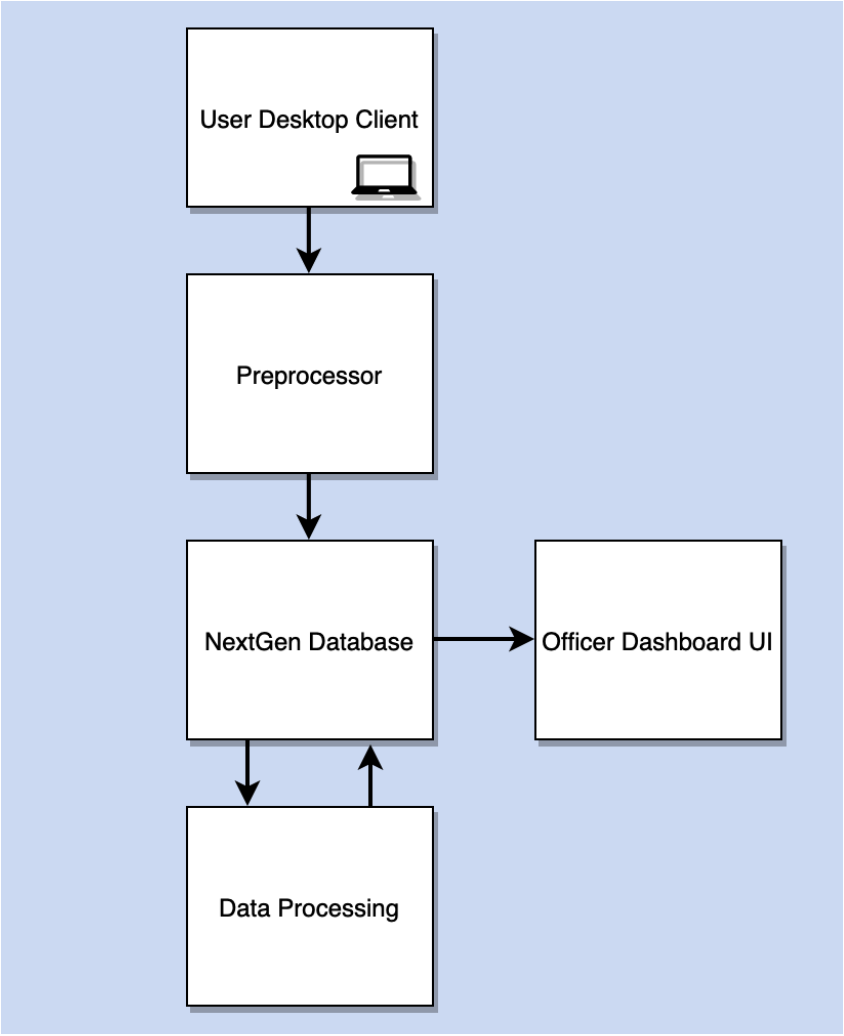
This project includes the set of components for monitoring devices and user activity.

Supported devices are Windows PC, Android devices, MacOS devices, iOS devices.

The main goal is to track user activity on devices, and then save to the database and show on the dashboard for officers.

Data Flow Schemas

Windows PC data tracking flow



User Desktop Client - Windows application which tracks user action on user machine and sends it to remote preprocessor

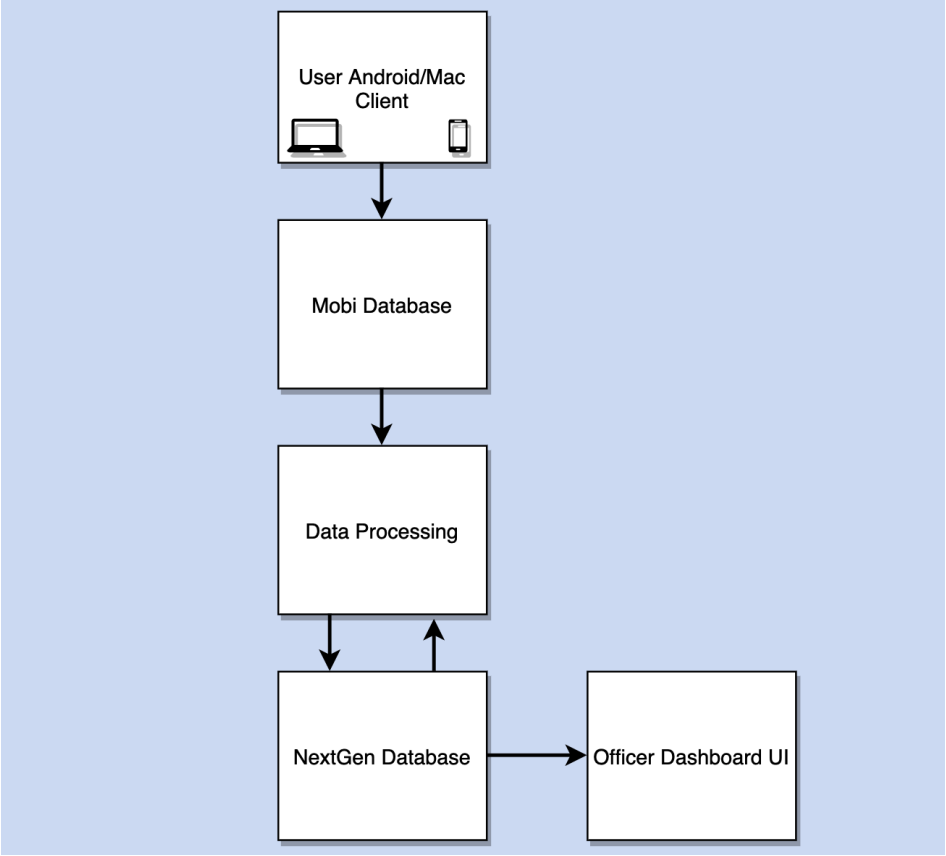
Preprocessor - gets data from User Desktop Client and prepares it to save this data to NextGen database

NextGen Database - MySQL database which stores all user activity data

Data Processing - set of applications to prepare and analyse raw user activity data before showing it on officer dashboard

Officer Dashboard UI - web application to show user activity for each registered device and make reports for user activity data.

Android/Mac data tracking flow



User Android/Mac Client - Windows application which tracks user action on user machine and sends it to remote preprocessor

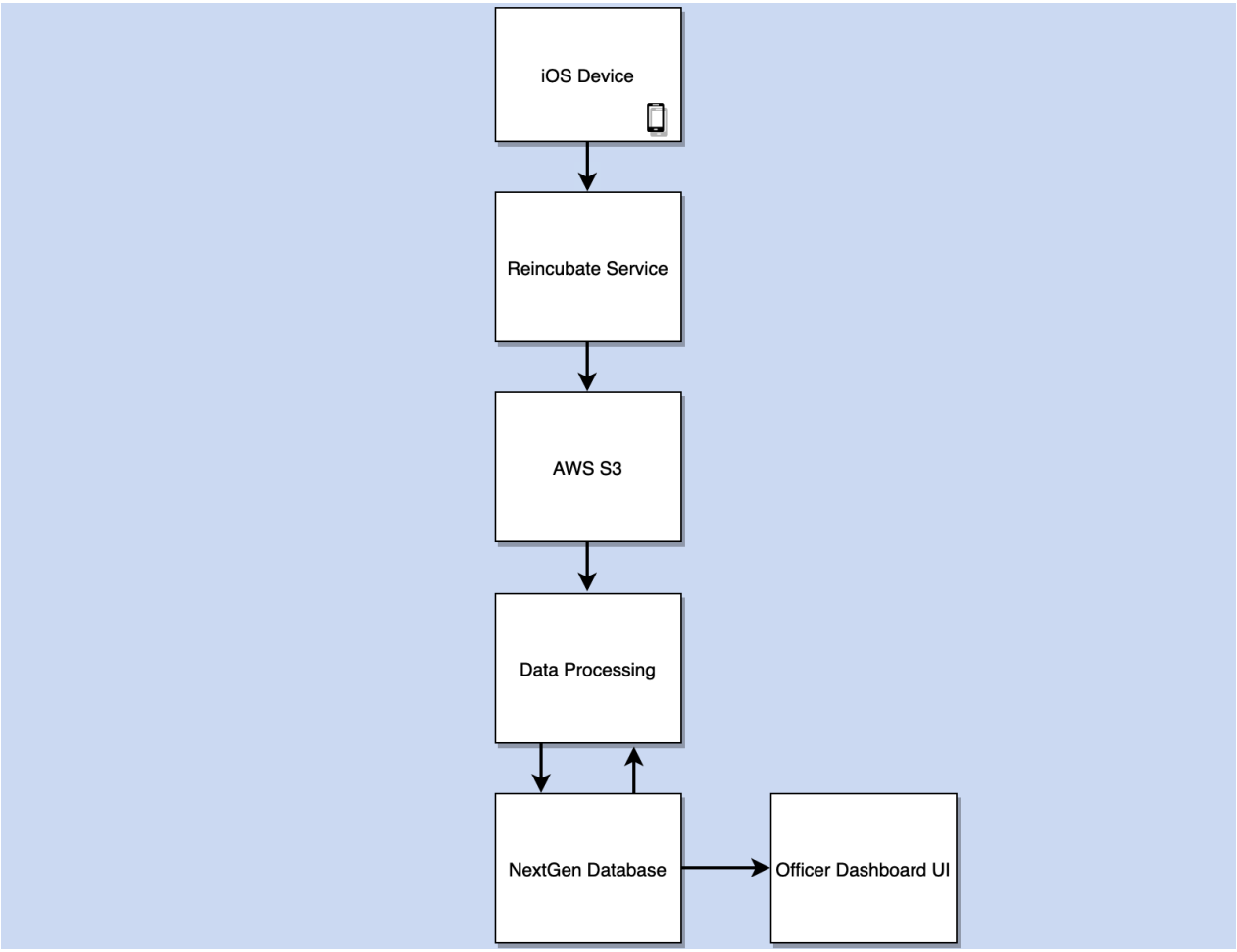
Mobi Database - MySQL database which stores all raw user activity data from device

NextGen Database - MySQL database which store all user activity data

Data Processing - set of applications to prepare and analyse raw user activity data before showing it on the officer dashboard.

Officer Dashboard UI - web application to show user activity for each registered device and make reports for user activity data.

iOS data tracking flow



iOS Device - iPhone/iPad where iCloud sync is enabled

Reincubate Service - 3rd party service (https://reincubate.com/ricloud-api/) gets data from iCloud and saves it to AWS S3 for registered devices

AWS S3 - file storage to save data from Reincubate.

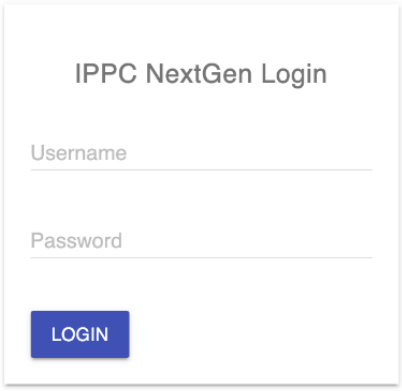
Data Processing - set of applications that prepare and analyze raw user activity data before displaying it on the officer dashboard. For iOS date processing, gets data from s3 and creates user activity data the in NextGen database.

NextGen Database - MySQL database which stores all user activity data.

Officer Dashboard UI - web application to show user activity for each registered device and generate reports for user activity data.

NextGen Dashboard UI Overview

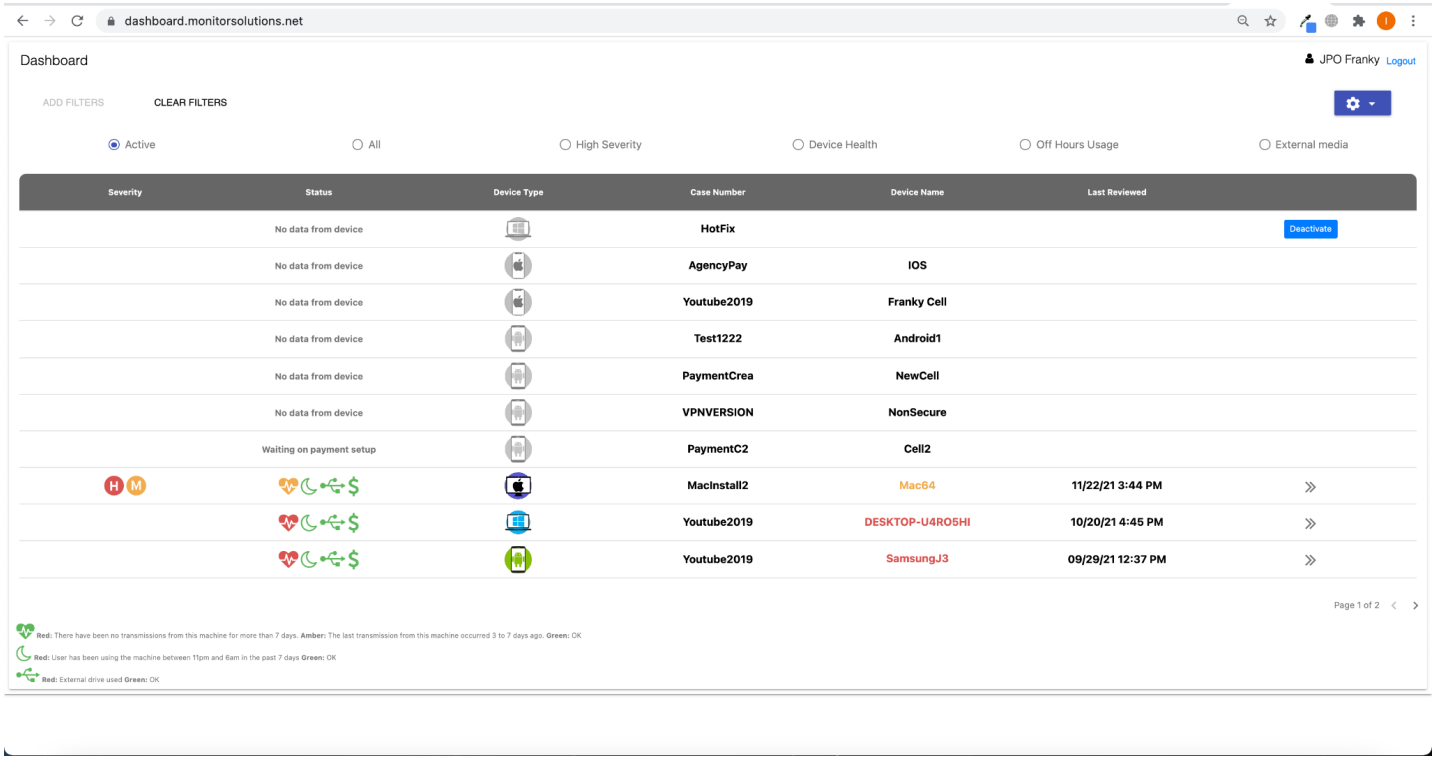
Login page



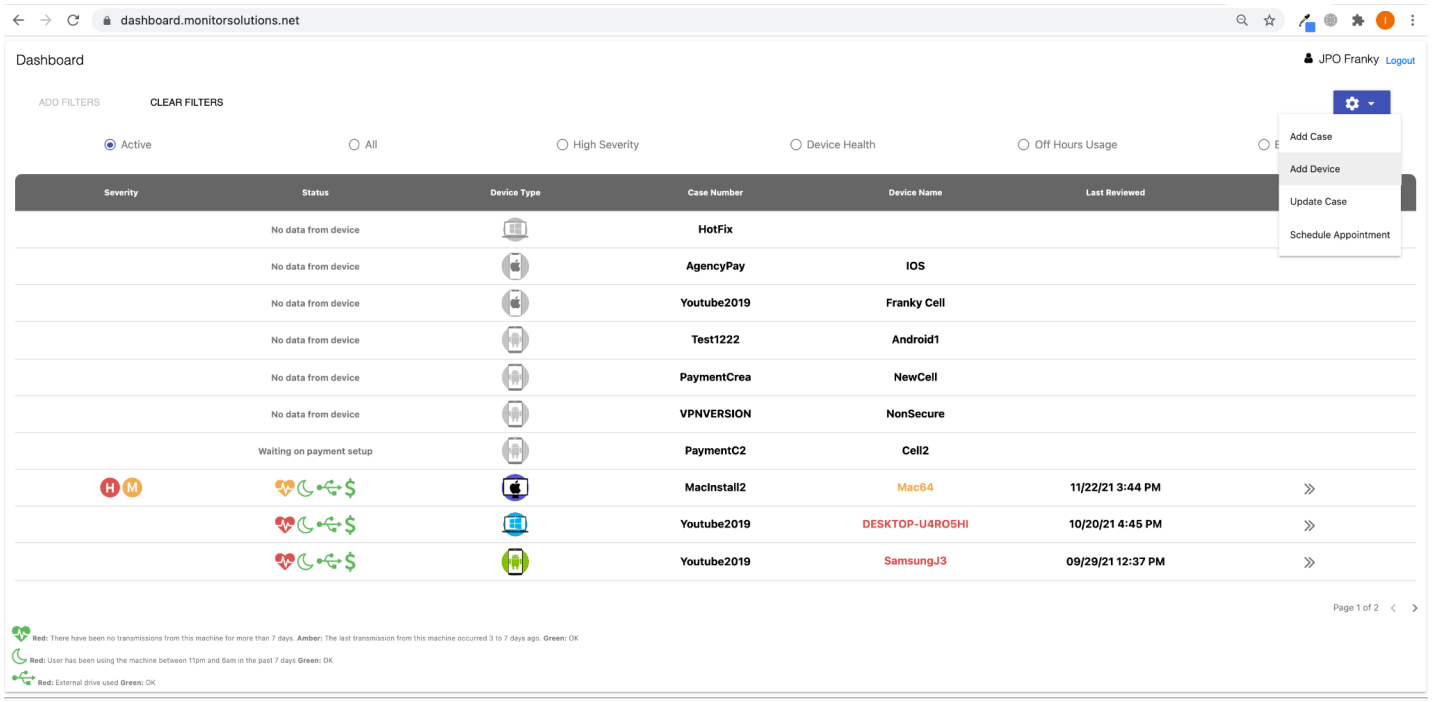
The Officer login to NextGen dashboard by login/password and redirected to Officer Dashboard page

Officer Dashboard page

Device List

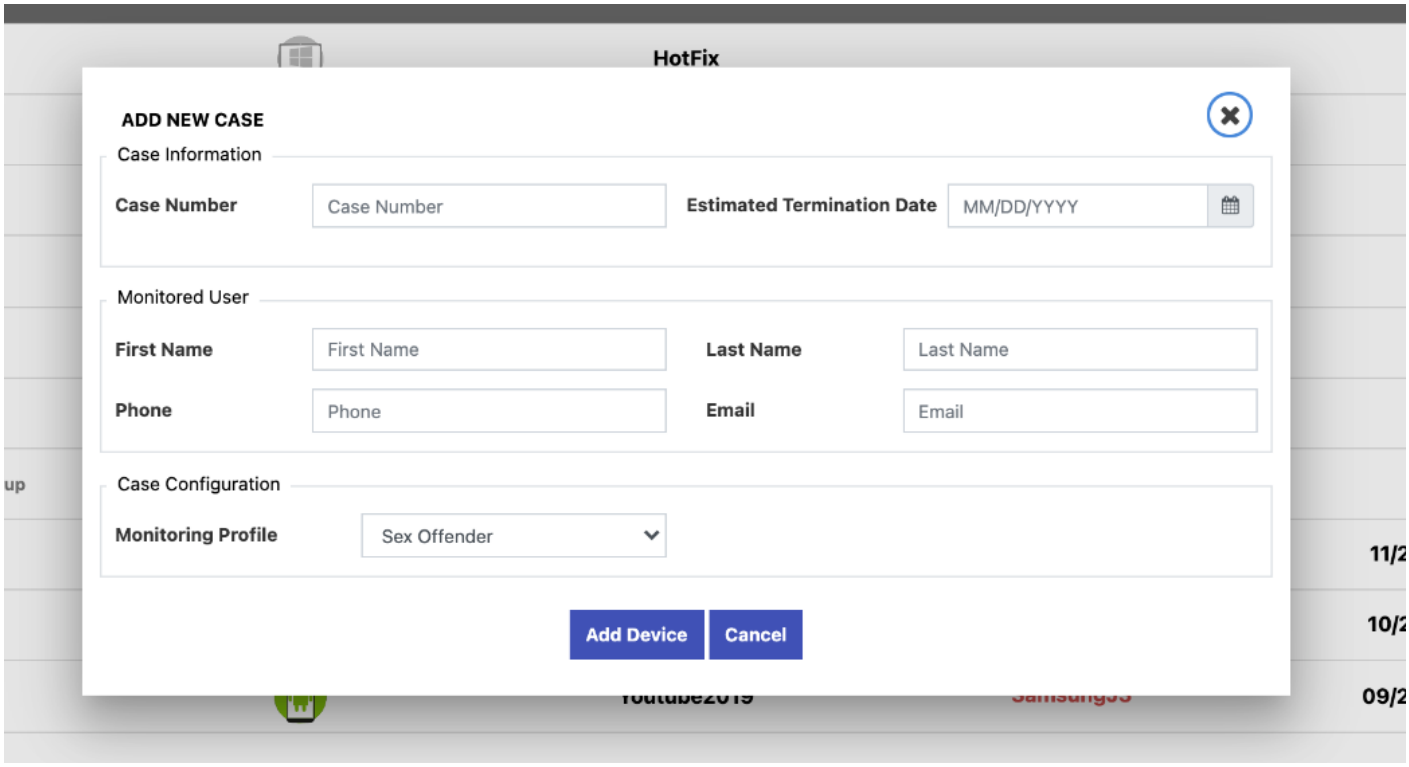


List of devices assigned to Officer



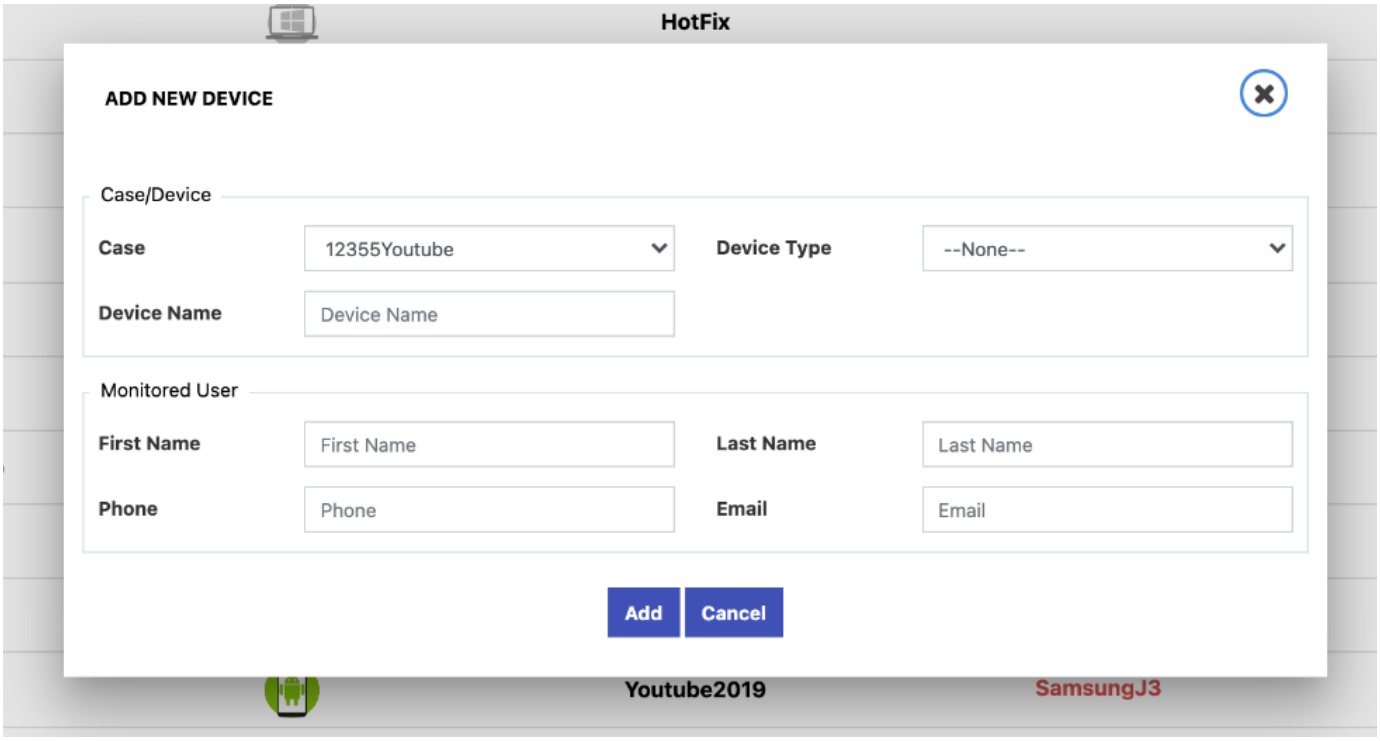
Button to create a new case

Create Case



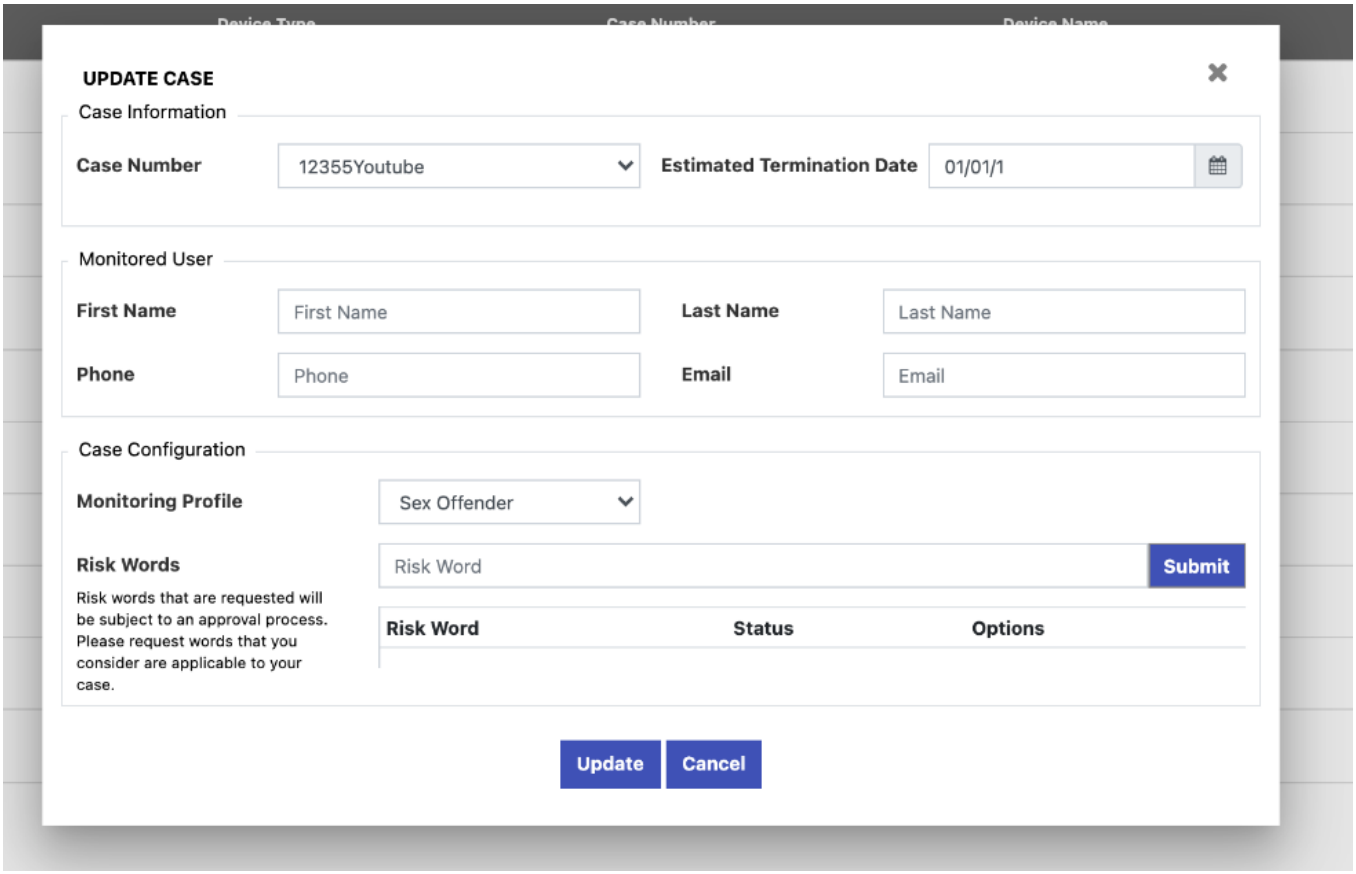
Create a new case dialog

Create Device



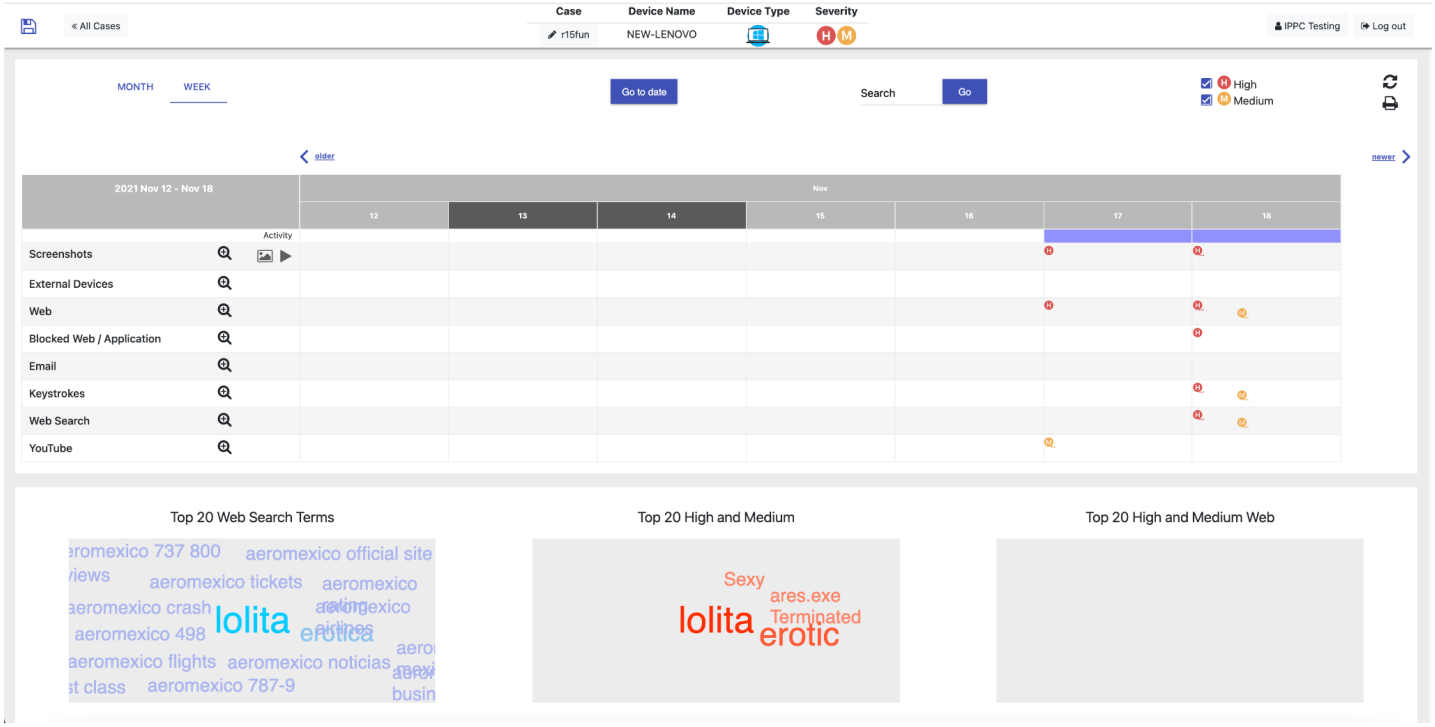
Create a new device dialog

Update Device



Update an existing device dialog

Device Dashboard

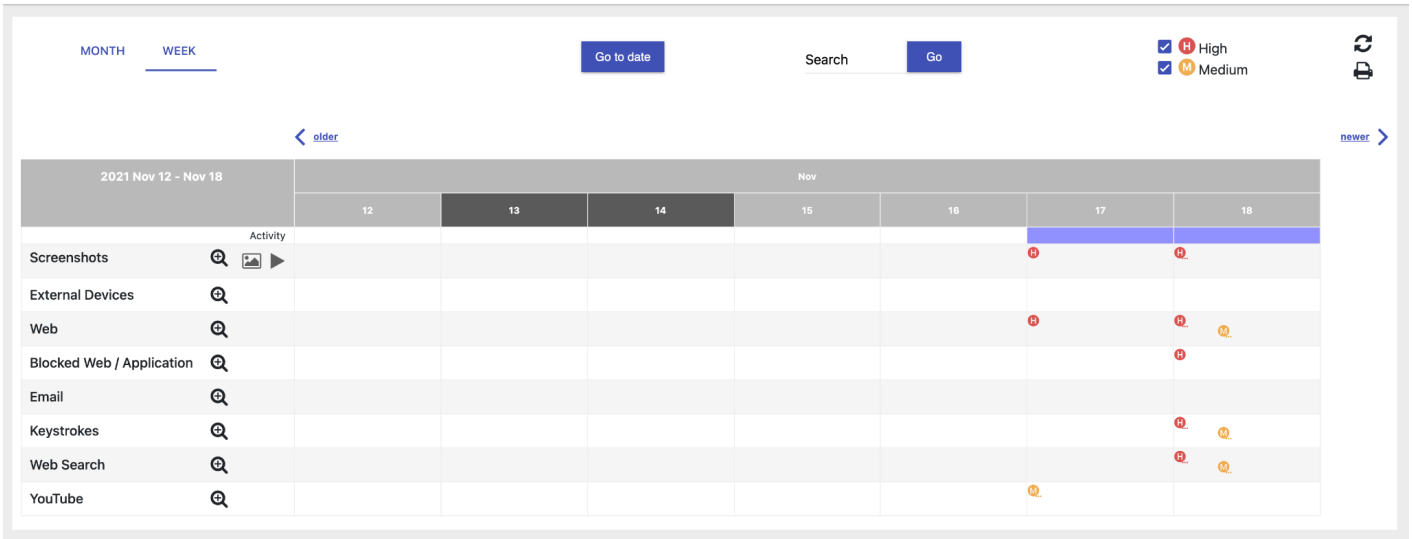


Device Details

Case	Device Name	Device Type	Severity
r15fun	NEW-LENOVO		H M

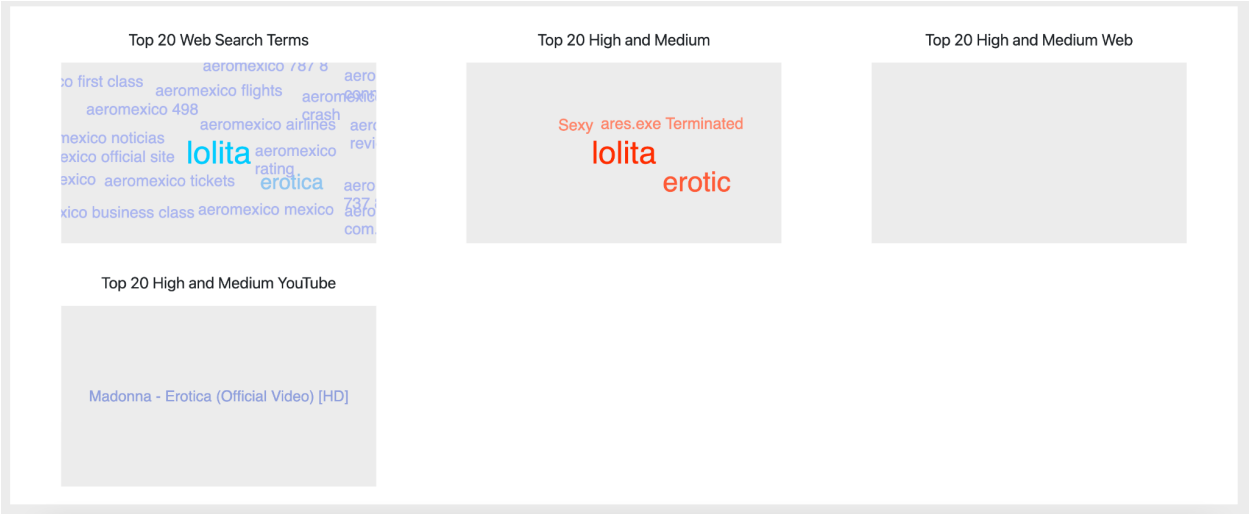
Device details information, Case, Device, Type and Severity

Device Events Swimlanes



Timelines with events by event types

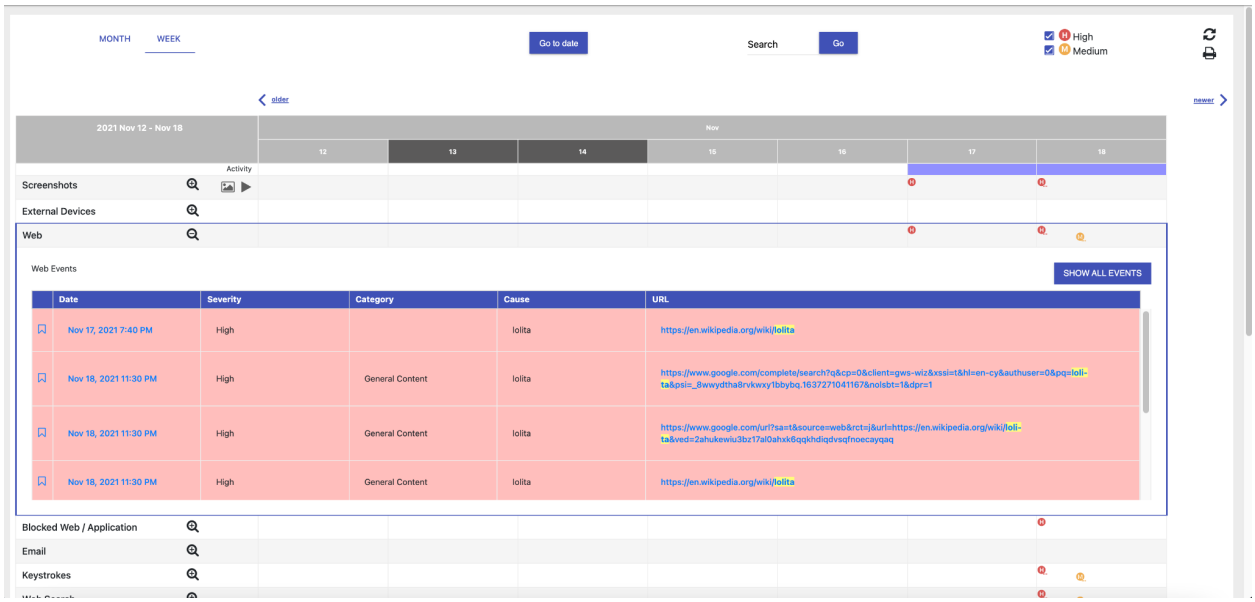
Tag Clouds



Most frequent words for web/web search/youtube events

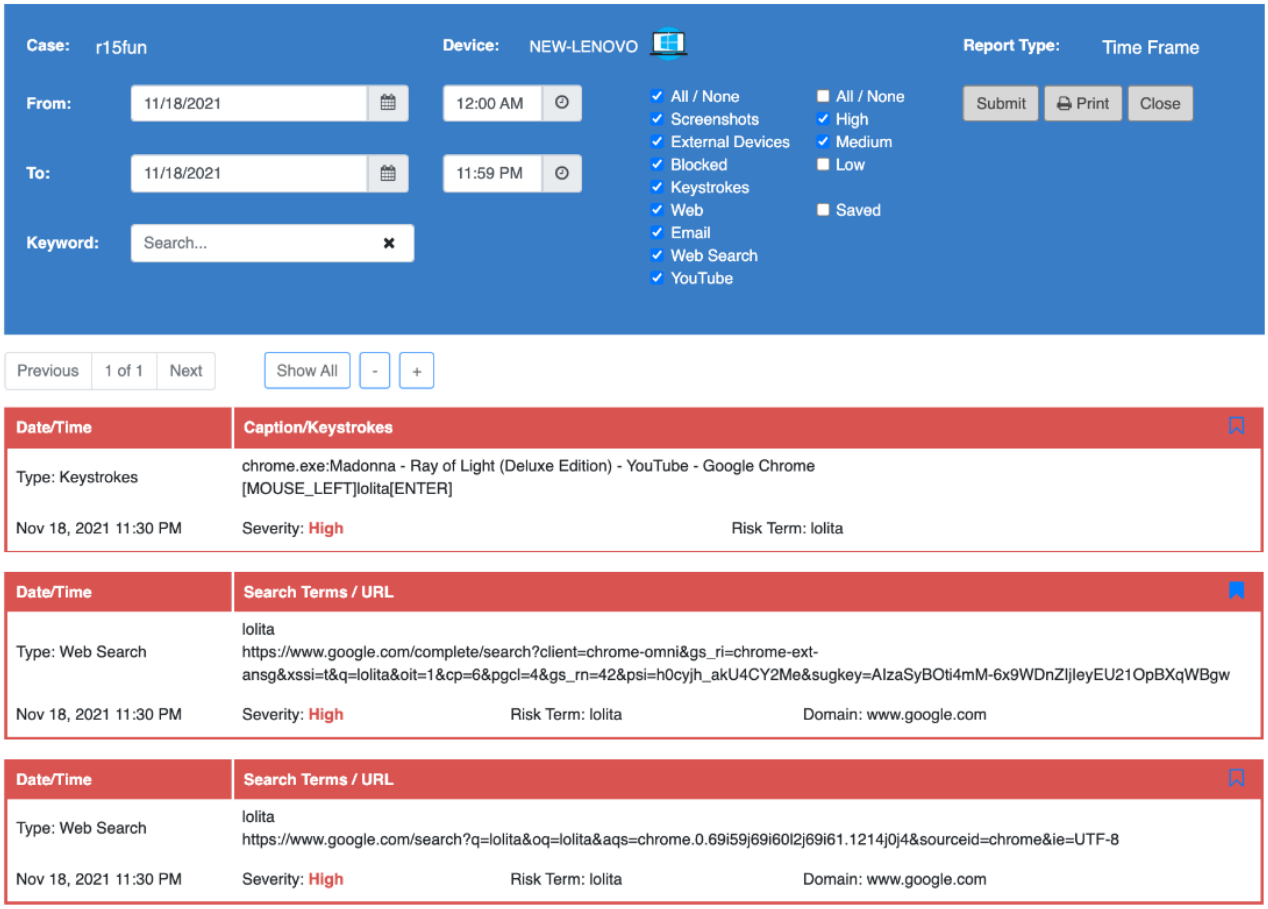
Reports

Expand Swimlane Report



Report gets a list of events for the time range for s particular event type (click to magnifier icon to open).

Timeframe Report



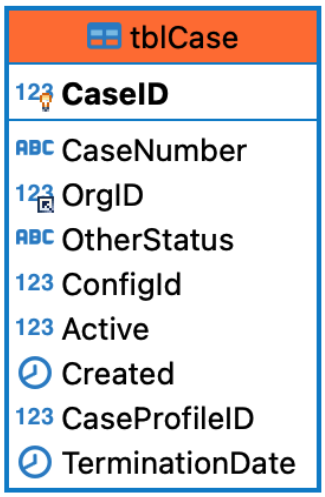
List of events in chronological order for the selected time range,event type and severity

Case/Device/Officer

For each user end device there are 3 main properties for system Case, Device and Officer

Case

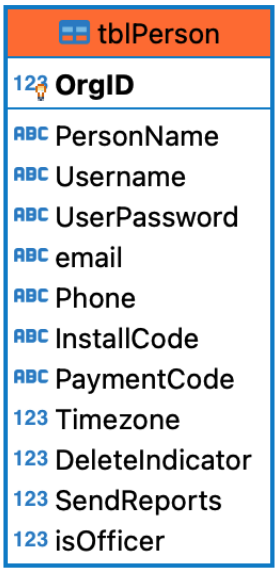
Case is a set of devices assigned to an officer. Case can contain several devices.
All cases store in **tblCase** in database



Officer

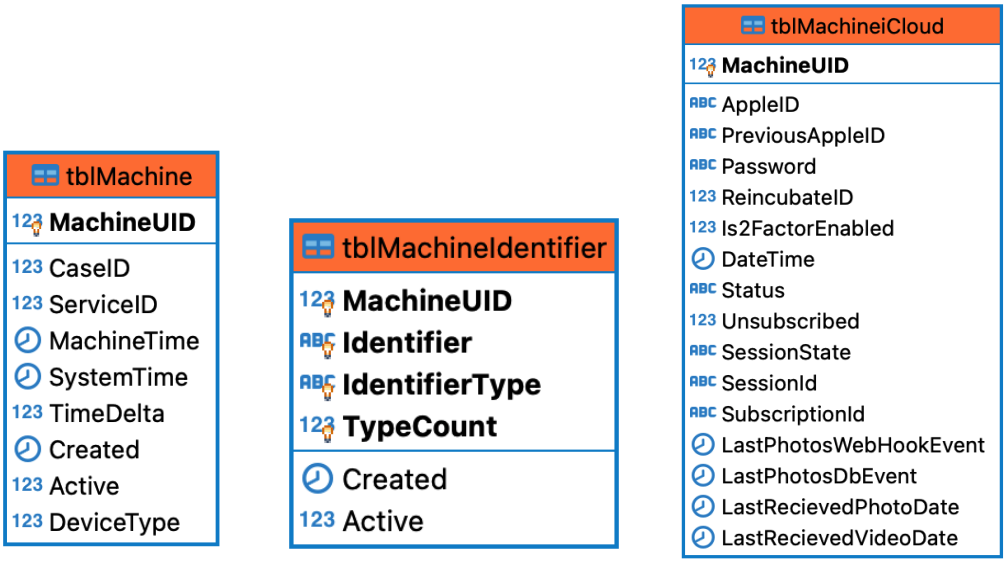
An officer is a person to whom a case is assigned. Officers can login to NexGen Dashboard and see a list of assigned cases. (Referenced by **OrgID** in **tblCase** table)

All officers stored in **tblPerson**



Device

The device is user end machine iOS, Windows PC, Android or MacOS
All information about devices is stored in the **tblMachine** and **tblMachineIdentifier** tables.
And in also **tblMachineiCloud** for iOS devices



Supported device types

Windows PC, iOS, Android and MacOS

ServiceID column in **tblMachine** define the type of device

0 - Windows PC
1 - iOS
2 - Android
3 - MacOS

Event

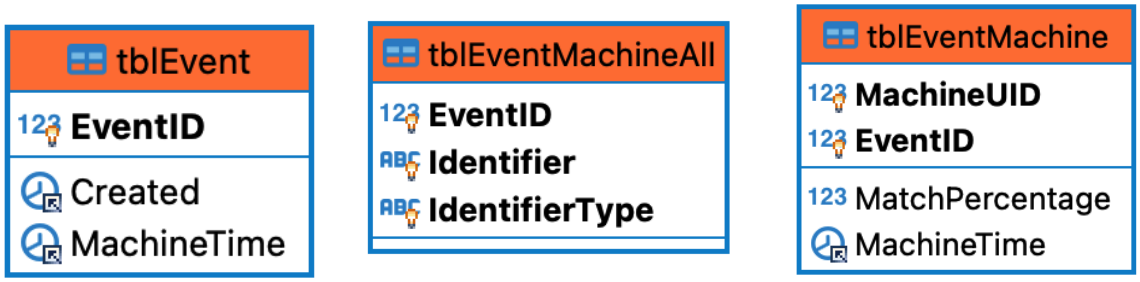
Event is user activity action on device (e.g. web site visiting, application run, photo, video)

Types of events

Type	Description	Device supported
Web	Common event for web site visiting	Windows, Android, Mac
WebSearch	Event of visiting web site with search query	Windows, Android, Mac
YouTube	Event of visiting youtube video url	Windows, Android, Mac
Screenshot	Screenshot of desktop screen or android device screen	Windows, Android, Mac
Photo	Just android or ios photo	Android, iOS
Blocked Web/App	Application or web url blocked by user agent on Window or Android device	Windows, Android
Keystrokes	Event of any keyboard activity	Windows, Android, Mac
Phone Call	Android phone call event	Android
SMS/MMS	Android phone sms or mms event	Android
Video	Just android or ios video	Android, iOS
Email	Email sended from windows device	Windows
External Device	Event of mount new external drive to windows	Windows

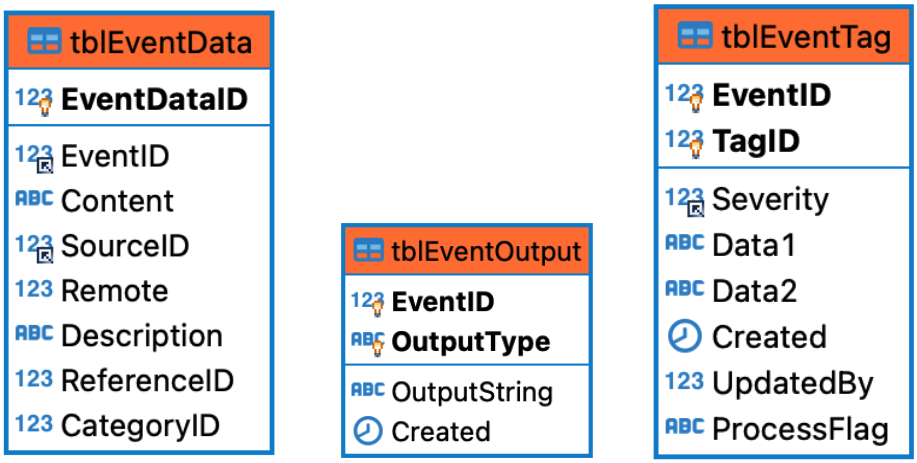
Database

The main tables to store event information include:



tblEvent, **tblEventMachineAll**, **tblEventMachine** - these tables contain information about events linked to the user device.

- For already identified events **MachineUID** field (from **tblEventMachine)** refereced to user device from **tblMachine**
- For not identified event Identifier and IdentifierType fields (from tblEventMachineAll) referenced to user device from **tblMachineIdentifier**



tblEventData, tblEventOutput, tblEventTag these tables contains information about

- event data - some event content (e.g web url, image url, some text etc),
- event type - combination of SourceID field (from tblEventData), OutputType (from tblEventOutput) and TagID (from tblEventTag)
- severity - 0 and 11 for Low, 5 for Medium and 10 for High

Sources Id

Windows

Data Type	SourceID	Swimlane
Admin	2	Advanced
Mail	4	Email
Newsgroup	5	
Web	6	Web
FTP	7	
Images	10	Screenshots
Keystrokes	11	Keystrokes
Web-Secure	12	Web
Files	30	Screenshots
Drives	31	Dashboard
Application Activity	32	
Health: Network	110	Advanced
Health: Config Request	120	Advanced
Health: Client Version	121	Advanced
Health: External IP	122	Advanced
Health: Local IP	123	Advanced
Health: MacAddress	124	Advanced

iOS

Data Type	SourceID	Swimlane
iCloud Images	20	Images
iWeb	22	Web
Calls	25	
iVideo	26	Video
iContact	27	
iCalendar	28	

Android

Data Type	SourceID	Swimlane
Camera	40	Multimedia
Keystrokes	41	Keystrokes
Web	42	Web
Screenshots	43	Screenshots
Calls	45	Reports
SMS	46	Messaging
MMS	47	Messaging
Video	49	Multimedia
GPS		Advanced

Mac

Data Type	SourceID	Swimlane
Screenshots	50	Screenshots
Keystrokes	51	Keystrokes
Web	52	Web

	Collecting Data but we are not using it
	Collecting Data but not collecting all the data
	Not Collecting Data

And mapping events by SourceID and TagID to UI components:

Table to be provided in Excel, Google Sheets or PDF format
IPPC/Epic Categories (Tags and Source ids)

Open documentation task - tblEventData/tblEventOutput fields explanation

Events UI

Open documentation task - UI screens

Rules

Rules are a fundamental part of the IPPC applications to identify possible alerts (events) that can be considered as dangerous. Actions on the offender devices are evaluated considering the rules defined on the following tables.

Tables

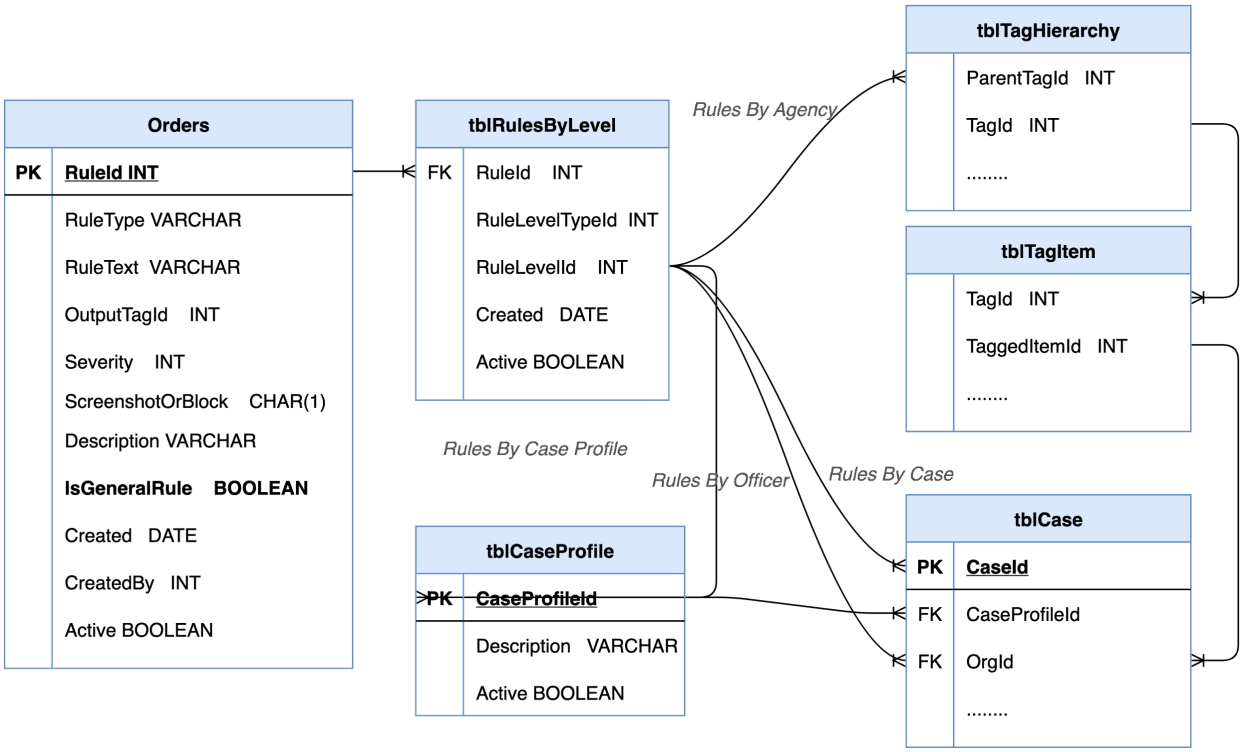
1. **tblRule**:
    a. Rules are defined here by type *(field: RuleType)*
    b. Rules can be defined as general for all the cases *(field: IsGeneralRule = true)*

2. **tblRulesByLevel**:
    a. Rules that are defined as specific *(field: IsGeneralRule = false)* can be assigned to the different levels. The levels are specified on field: *RuleLevelTypeId* (1 = Case Profile, 2 = Agency, 3 = Officer, 4 = Case)
    b. The field *RuleLevelId* is a “dynamic forean key”. It is the Id of the case profile, agency, officer or case. So for example: *RuleId* = 5, *RuleLevelTypeId = 4 and RuleLevelId = 87,* it means that the rule number 5 is a rule specific for a case *(RuleLevelTypeId = 4*) and the case id is the number 87.

3. **tblCaseProfile:**
    a. This table contains the different case profiles that IPPC manages (sex offender, ciber crimilnal, etc..)

4. **tblTagHierarchy** and **tblTagItem**
    a. These tables contain the structure of agencies, supervisors and officers. The logic of “rules by agency” passes over this structure getting all the officers that belong to an agency and then bring all the cases that belong to these officers using the field *OrgId* on table *tblCase.*

5. **tblCase**
    a. This table case has the fields CaseId, OrgId and CaseProfileId that are the key fields to link the rules through the different levels.


Orders
PK RuleId INT
RuleType VARCHAR
RuleText VARCHAR
OutputTagId INT
Severity INT
ScreenshotOrBlock CHAR(1)
Description VARCHAR
IsGeneralRule BOOLEAN
Created DATE
CreatedBy INT
Active BOOLEAN
tblRulesByLevel
FK RuleId INT
RuleLevelTypeId INT
RuleLevelId INT
Created DATE
Active BOOLEAN
Rules By Agency
Rules By Case Profile
Rules By Officer
Rules By Case
tblTagHierarchy
ParentTagId INT
TagId INT
.....
tblTagItem
TagId INT
TaggedItemId INT
.....
tblCaseProfile
PK CaseProfileId
Description VARCHAR
Active BOOLEAN
tblCase
PK CaseId
FK CaseProfileId
FK OrgId
.....

Stored Procedures

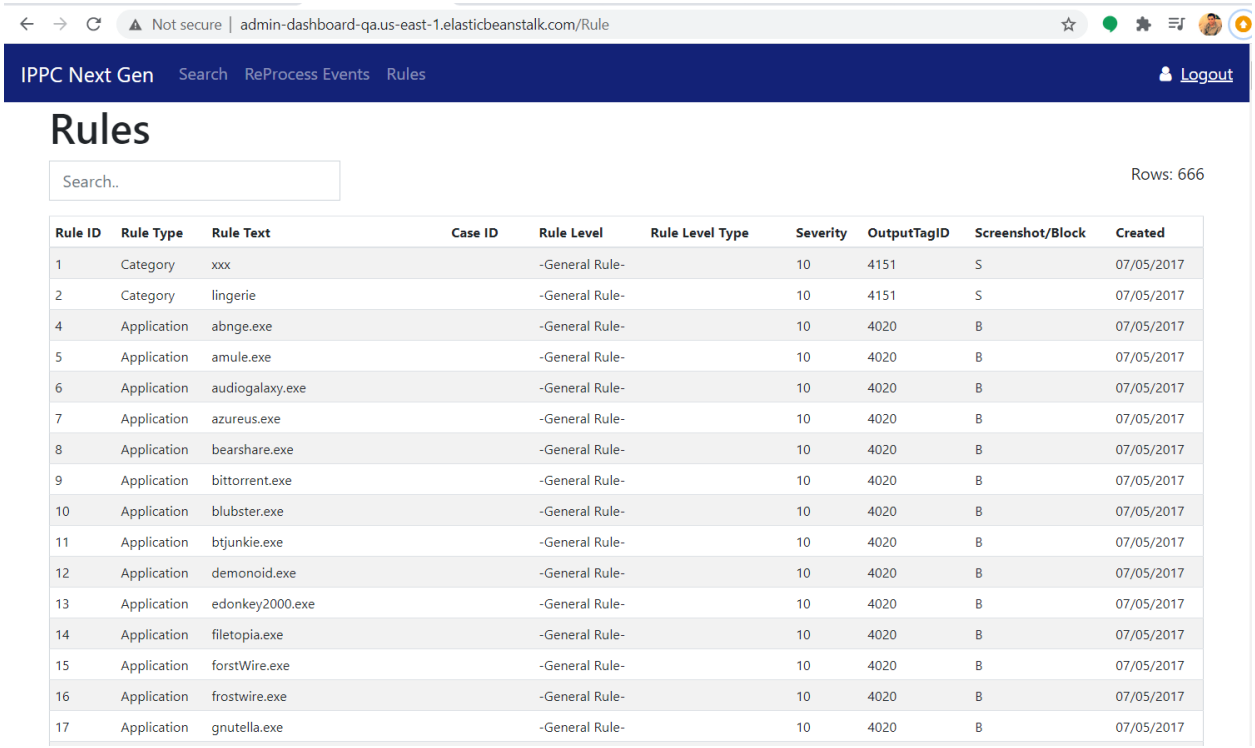
1. **pRuleAllGet**
    a. This stored procedure brings all the rules, general and specific. Those rules with *CaseId* == *null.* Are considered general and the rest of them are rules specific for a level (Case Profile, Officer, Agency or Case).

2. **pRuleCaseGet**
    a. This stored procedure brings all the rules, general and specific for a *CaseId*.
    b. The logic of this stored procedure is the same that ***pRuleAllGet***, the difference is that this one brings only the rules that apply for the *CaseId*.

Rules on Admin Dashboard tool

The Admin Dashboard provides a screen dedicated to consult the existing Rules. This screen uses the stored procedure: ***pRuleDescriptionAllGet*** to get all the rules. This is the same logic that the previous stored procedures use to get all the rules, but this one brings more information to readable for IPPC personnel.

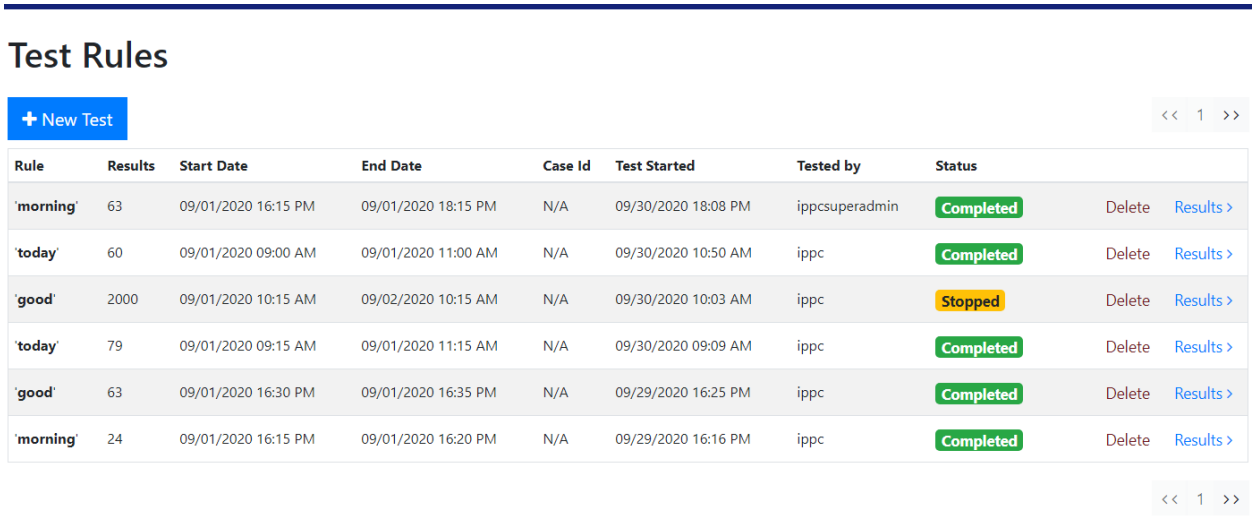
This screen allows users to search the rules by the information that is shown on it.



Rule ID	Rule Type	Rule Text	Case ID	Rule Level	Rule Level Type	Severity	OutputTagID	Screenshot/Block	Created
1	Category	xxx		-General Rule-		10	4151	S	07/05/2017
2	Category	lingerie		-General Rule-		10	4151	S	07/05/2017
4	Application	abnge.exe		-General Rule-		10	4020	B	07/05/2017
5	Application	amule.exe		-General Rule-		10	4020	B	07/05/2017
6	Application	audiogalaxy.exe		-General Rule-		10	4020	B	07/05/2017
7	Application	azureus.exe		-General Rule-		10	4020	B	07/05/2017
8	Application	bearshare.exe		-General Rule-		10	4020	B	07/05/2017
9	Application	bittorrent.exe		-General Rule-		10	4020	B	07/05/2017
10	Application	blubster.exe		-General Rule-		10	4020	B	07/05/2017
11	Application	btjunkie.exe		-General Rule-		10	4020	B	07/05/2017
12	Application	demonoid.exe		-General Rule-		10	4020	B	07/05/2017
13	Application	edonkey2000.exe		-General Rule-		10	4020	B	07/05/2017
14	Application	filetopia.exe		-General Rule-		10	4020	B	07/05/2017
15	Application	forstWire.exe		-General Rule-		10	4020	B	07/05/2017
16	Application	frostwire.exe		-General Rule-		10	4020	B	07/05/2017
17	Application	gnutella.exe		-General Rule-		10	4020	B	07/05/2017

Testing new Rules on Admin Dashboard tool

In the Admin Dashboard there is a screen called "***Test Rules***" that allows to test a new possible rule. The purpose of this screen is to test a keyword that can be defined as a new rule.



Test Rules

+ New Test

<< 1 >>

Rule	Results	Start Date	End Date	Case Id	Test Started	Tested by	Status		
'morning'	63	09/01/2020 16:15 PM	09/01/2020 18:15 PM	N/A	09/30/2020 18:08 PM	ippcsuperadmin	Completed	Delete	Results >
'today'	60	09/01/2020 09:00 AM	09/01/2020 11:00 AM	N/A	09/30/2020 10:50 AM	ippc	Completed	Delete	Results >
'good'	2000	09/01/2020 10:15 AM	09/02/2020 10:15 AM	N/A	09/30/2020 10:03 AM	ippc	Stopped	Delete	Results >
'today'	79	09/01/2020 09:15 AM	09/01/2020 11:15 AM	N/A	09/30/2020 09:09 AM	ippc	Completed	Delete	Results >
'good'	63	09/01/2020 16:30 PM	09/01/2020 16:35 PM	N/A	09/29/2020 16:25 PM	ippc	Completed	Delete	Results >
'morning'	24	09/01/2020 16:15 PM	09/01/2020 16:20 PM	N/A	09/29/2020 16:16 PM	ippc	Completed	Delete	Results >

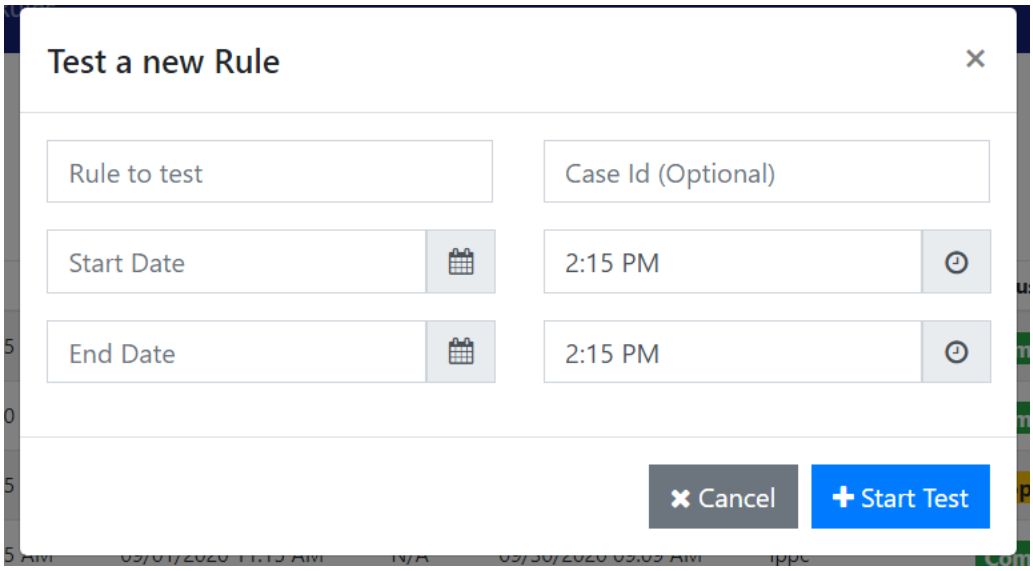
<< 1 >>

Adding a new Test

Once a new test is created, a new process is started on the Web API called: **IPPC.Process.API**

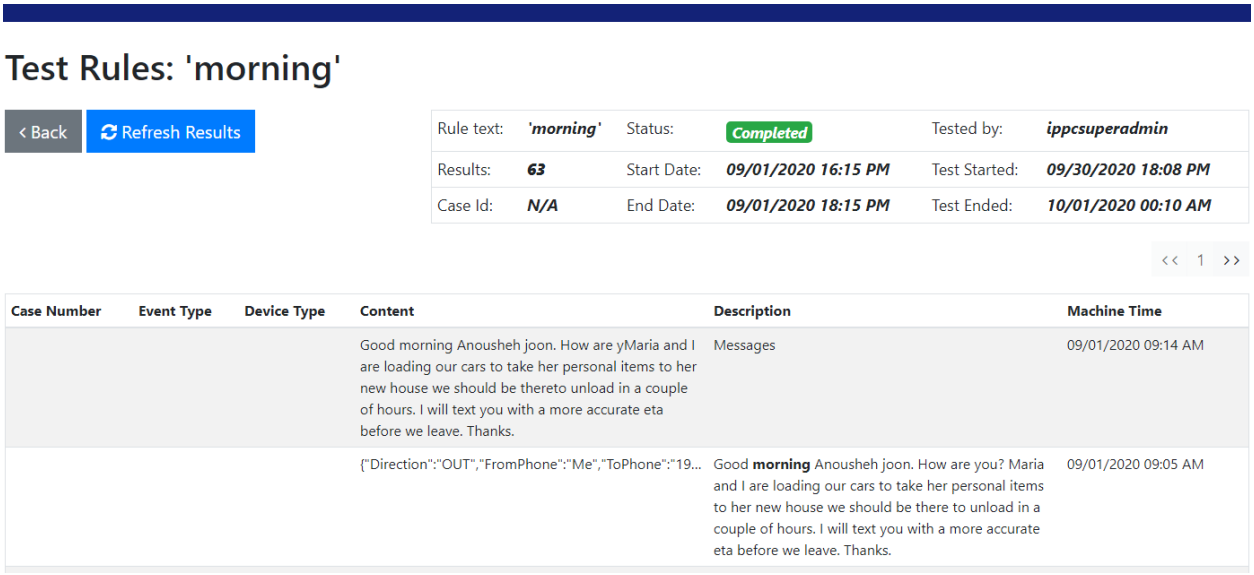
This API has a test method:

QA: http://processing-webapi-qa.us-east-1.elasticbeanstalk.com/api/TestAPI/Connect
PROD: http://processing-webapi-prod.us-east-1.elasticbeanstalk.com/api/TestAPI/Connect



Results of the Test

Clicking on “Results” to review the events where the work was found.



Test Rules: 'morning'

‹ Back | Refresh Results

Rule text:	*'morning'*	Status:	*Completed*	Tested by:	*ippcsuperadmin*
Results:	*63*	Start Date:	*09/01/2020 16:15 PM*	Test Started:	*09/30/2020 18:08 PM*
Case Id:	*N/A*	End Date:	*09/01/2020 18:15 PM*	Test Ended:	*10/01/2020 00:10 AM*

<< 1 >>

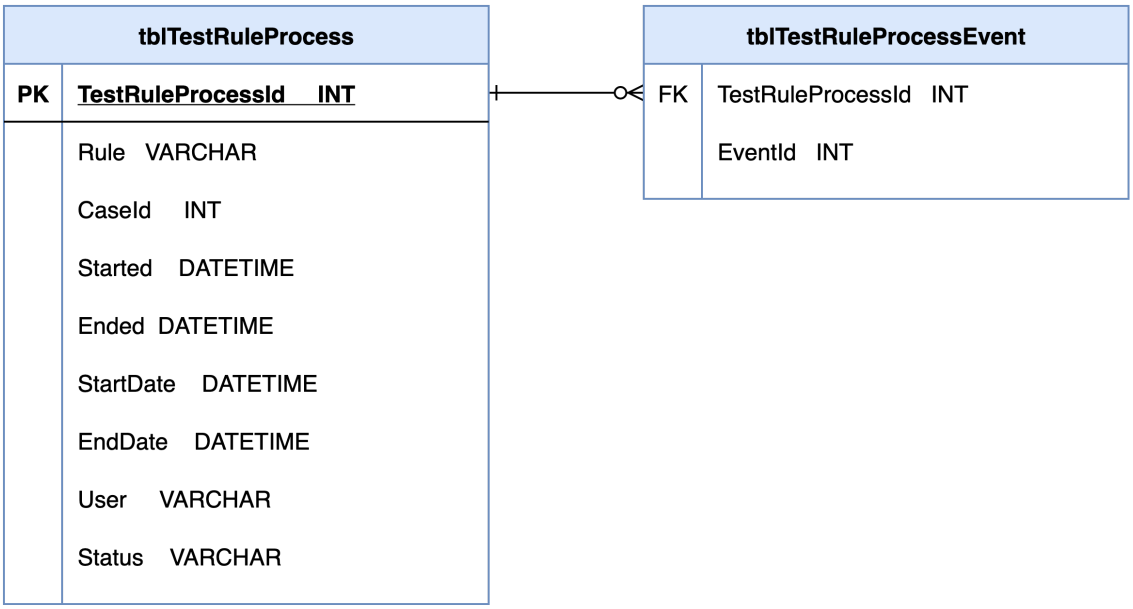
Case Number	Event Type	Device Type	Content	Description	Machine Time
			Good morning Anousheh joon. How are yMaria and I are loading our cars to take her personal items to her new house we should be thereto unload in a couple of hours. I will text you with a more accurate eta before we leave. Thanks.	Messages	09/01/2020 09:14 AM
			{"Direction":"OUT","FromPhone":"Me","ToPhone":"19...	Good **morning** Anousheh joon. How are you? Maria and I are loading our cars to take her personal items to her new house we should be there to unload in a couple of hours. I will text you with a more accurate eta before we leave. Thanks.	09/01/2020 09:05 AM

Testing Rules Tables

The Rule testing process involves two tables:

1. **tblTestRuleProcess**:
    a. This table contains the criteria to execute the test.

2. **tblTestRuleProcessEvent**:
    a. This table contains the events that matched with the search criteria.

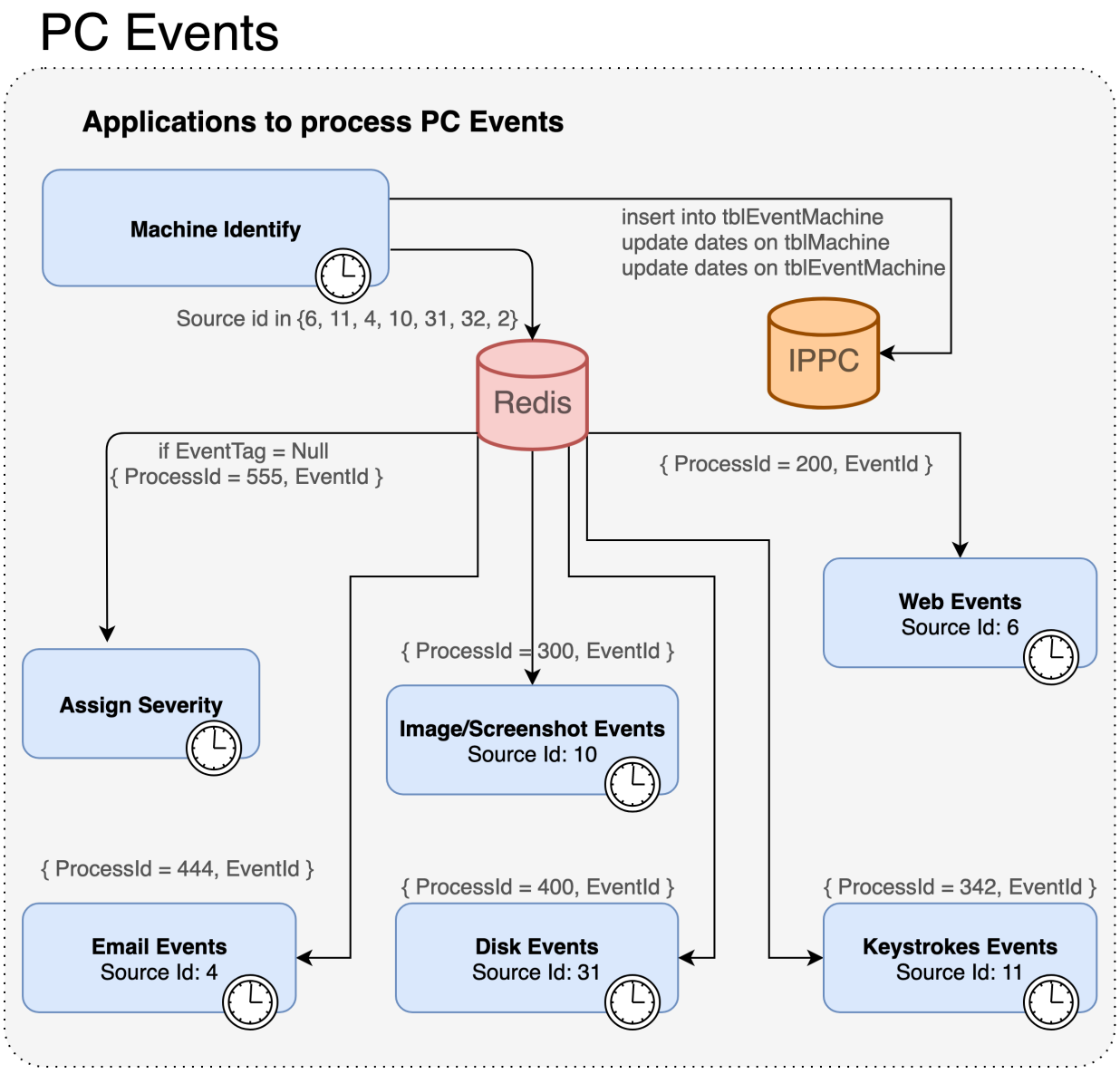


Processing Overview

Processing PC Events

All PC Events are created in the NextGen database by Legacy preprocessor.

The starting point to process PC events is the "***Machine Identify***" application with the AWS EB profile: "***machine-identify-processing***". This application interacts with the Redis Queues, it evaluates the Source Id and adds the events to the proper queue indicating the process Id and the event id.



PC Device Identify Worker

This worker creates a reference between the event and the user-end machine in the database.
Insert new record to **tblEventMachine** table and put event to appropriate redis queue (depends of event type) for next processing steps

VS Project: MachineIdentifyEventsProcessing.Worker
EB Environment: machine-identify-processing-prod,machine-identify-processing-qa
Logs: NextGen.Processing.PC.QA, NextGen.Processing.PC.Production

PC Assign Severity Worker

This worker handles PC screenshots and blocked app runs events.
It applies rules logic for screenshot trigger annotations.

Input redis queue: 555
VS Project: AssignSeverityProcessing.Worker
EB Environment: assign-serverity-processing-qa,assign-serverity-processing-prod
Logs: NextGen.Processing.PC.QA, NextGen.Processing.PC.Production

PC Email Worker

This worker handles PC emails events.
It applies rules logic for email text and subject and extract email attachments.

Input redis queue: 444
VS Project: PCEmailEventsProcessing.Worker
EB Environment: pc-email-processing1-qa,pc-email-processing1-prod
Logs: NextGen.Processing.PC.QA, NextGen.Processing.PC.Production

PC Image Worker

This worker handles PC screenshots events.
It makes UUDecode for screenshots and put to next redis queue for advanced image analysis

Input redis queue: 300
VS Project: PCImageEventsProcessing.Worker
EB Environment: pc-image-processing-qa,pc-image-processing-prod
Logs: NextGen.Processing.PC.QA, NextGen.Processing.PC.Production

PC Disk Worker

This worker handles PC external drive events to detect plug and unplug new external drives to PC

Input redis queue: 400
VS Project: PCImageEventsProcessing.Worker
EB Environment: pc-disk-processing-qa,pc-disk-processing-prod
Logs: NextGen.Processing.PC.QA, NextGen.Processing.PC.Production

PC Web Worker

This worker handles PC Web/WebSearch/YouTube events.
It determines which web event comes from Web,WebSearch or YouTube and apply rules logic for url and content.

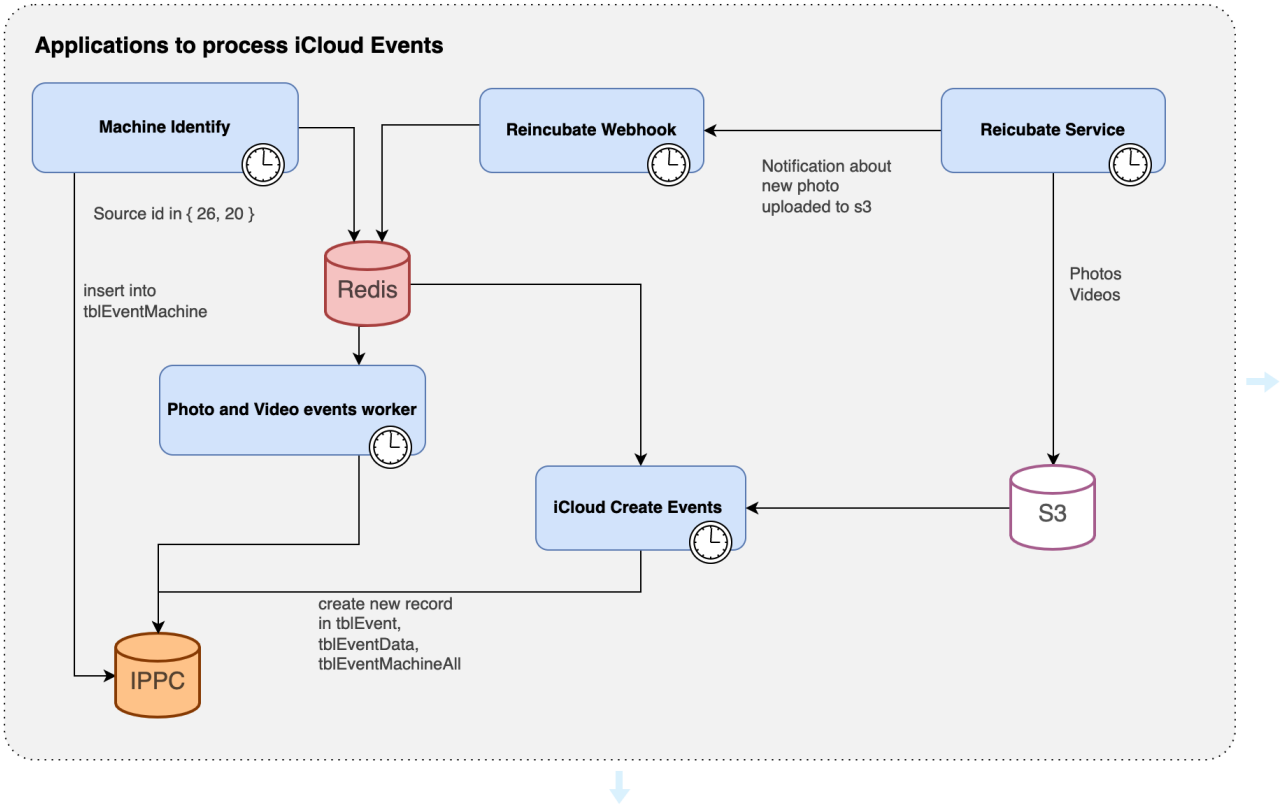
Input redis queue: 200
VS Project: PCWebEventsProcessing.Worker
EB Environment: pc-web-processing-qa,pc-web-processing-prod
Logs: NextGen.Processing.PC.QA, NextGen.Processing.PC.Production

PC Keystrokes Worker

This worker handles PC Keystrokes events.
It applies rules logic for keystroke text.

Input redis queue: 342
VS Project: PCKeystrokeEventsProcessing.Worker
EB Environment: pc-keystroke-processing-qa,pc-keystroke-processing-prod
Logs: NextGen.Processing.PC.QA, NextGen.Processing.PC.Production

Processing iOS Events



Reincubate Service

This is 3rd party service https://reincubate.com/ricloud-api/ which sends to iCloud data(photos and videos) to s3 and makes a call to webhook endpoint

Reincubate WebHook API

API which Reincubate Service call every time when new data upload to s3
This API save reference between iCloud account and s3 data to redis queue

VS Project: ReincubateWebHook.Web
EB Environment: icloud-processing-webhook-prod,icloud-processing-webhook-qa
Logs: NextGen.ReincubateWebHook..QA, NextGen.ReincubateWebHook.Production

iCloud Create Events Worker

This worker get photos and videos from redis and S3 and create events in NextGen database

Input redis queue: Reincubate_Polls_Queue
VS Project: iCloudEventsCreationProcessing.Worker
EB Environment: picloud-events-creation1-qa,icloud-events-creation1-prod
Logs: NextGen.Processing.iCloud.QA, NextGen.Processing.iCloud.Production

Device Identify Worker

This worker creates a reference between the event and the user-end machine in the database.
,insert new record to **tblEventMachine** table and put event to appropriate redis queue (depends of event type) for next processing steps

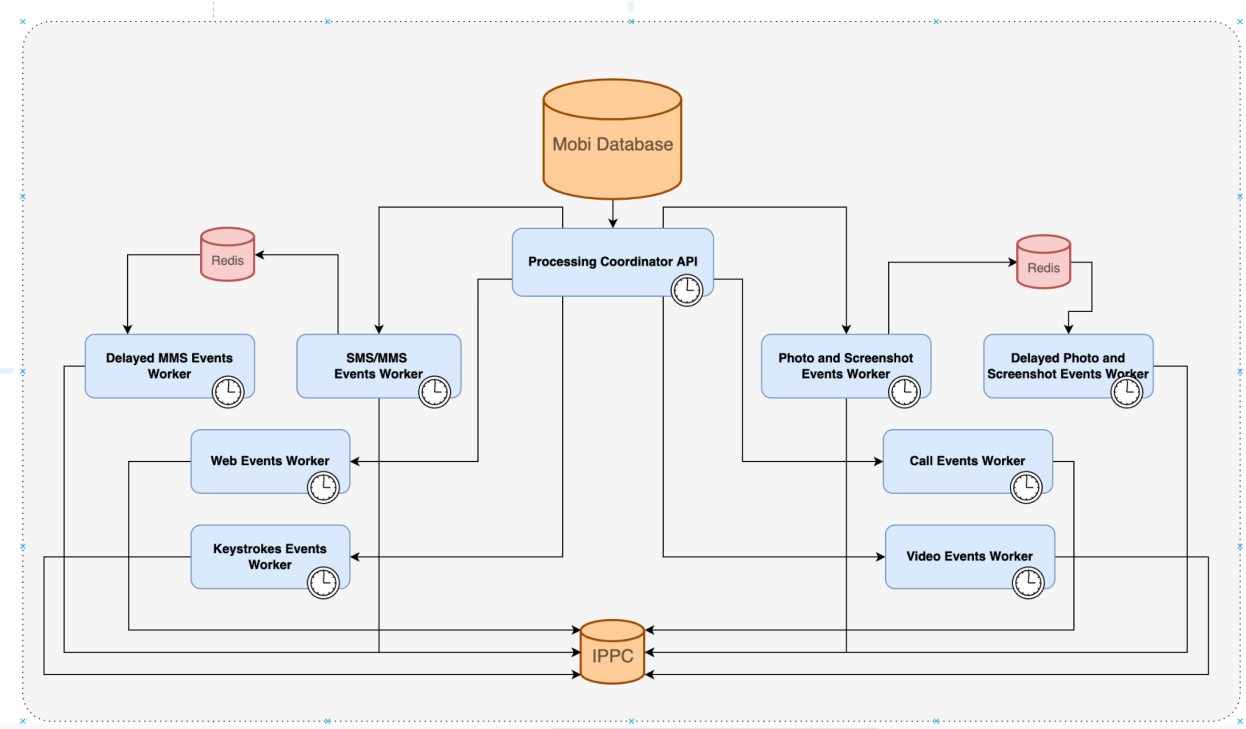
VS Project: MachineIdentifyEventsProcessing.Worker
EB Environment: machine-identify-processing-prod,machine-identify-processing-qa
Logs: NextGen.Processing.PC.QA, NextGen.Processing.PC.Production

iCloud Image Worker

Same instance as for PC
This worker handles ios photos and videos events and put photos to next redis queue for advanced image analysis

Input redis queue: 300
VS Project: PCImageEventsProcessing.Worker
EB Environment: pc-image-processing-qa,pc-image-processing-prod
Logs: NextGen.Processing.PC.QA, NextGen.Processing.PC.Production

Processing Android and Mac Events



The Android/Mac applications share the same code. The logic evaluates the SourceID and analyzes the event according to it. The goal of this analysis is to determine the severity of the events and complete the information about them.

The source of the Android/Mac events is the Mobi database. Our Android and Mac workers get information about the events from Mobi and create events in the NextGen database.

To sync getting events from Mobi by multiple workers without overlapping the event ranges we are using Processing Coordinator API web service which returns the next available event range for the Mobi table.

Each event type stored in Mobi database in a separate table

Type of events	Table to store
Android Call	log_cal_calls_1
Android/Mac Web	log_web_history_1
Android SMS/MMS	log_sms_text_message_1
Android/Mac Photo/Screenshot	log_pic_picture_1
Android/Mac Keys	log_act_activity_1

Visual Studio Project	Description	AWS EB Profile	Cloud Watch Log Name
ProcessingCoordinator.Web	API to sync getting events for multiple workers from Mobi database. Return new available event range.	processing-coordinator-prod2/processing-coordinator-qa1	NextGen.ProcessingCoordinator.QA/Production
Process**Call**AndroidMacEvents.Worker	Process call events	android-mac-linux-call	NextGen.Processing.Android.QA/Production
Delay**ScreenshotImages**Processing.Worker	Handle some delayed screenshots and photos.	delayed-screenshot-images-processing	NextGen.Processing.Android.QA/Production
Process**Image**AndroidMacEvents.Worker	Process photo and screenshot events	android-mac-linux-image-qa	NextGen.Processing.Android.QA/Production
Process**Key**AndroidMacEvents.Worker	Process keystroke events	android-mac-linux-key	NextGen.Processing.Android.QA/Production
Delay**MMSImages**Processing.Worker	Handle some delayed MMS with photos.	delayed-mms-images-processing	NextGen.Processing.Android.QA/Production
Process**SMS**AndroidMacEvents.Worker	Process SMS events	android-mac-linux-sms	NextGen.Processing.Android.QA/Production
Process**Video**AndroidMacEvents.Worker	Process video events	android-mac-linux-video	NextGen.Processing.Android.QA/Production
Process**Web**AndroidMacEvents.Worker	Process web events	android-mac-linux-web	NextGen.Processing.Android.QA/Production

Specific processing details

UUDecode

This is a process to get screenshots from PC machines.
The preprocessor sends screenshots from Windows PC device to S3 bucket as base64 encoded text file with .uue extension and event id in filename ([EVENTID].uue e.g. 23134356778.uue).
The UUDecode process decodes the base64 file and saves it as a jpg file to another s3 folder.

YouTube

The YouTube process is part of the web processing events worker for PC,Android and MacOS events.
If a web event contains “youtube.com” in the request url we start the youtube process for this event.

1. We get id of youtube video from url
2. Make request with youtube id to google api to get video title and thumbnail
3. Save video title and thumbnail to database
4. Apply text rules logic to video title to assign severity for this events

PC Email

It is a process to get emails from PC machines.
The preprocessor sends email data from Windows PC device to S3 bucket as file with .TMP extension and event id in filename ([EVENTID].uue e.g. 23134356778.TMP).
The PCEmails use the MimeKit library to get the subject, email body and attachments from this file and save it to the database.

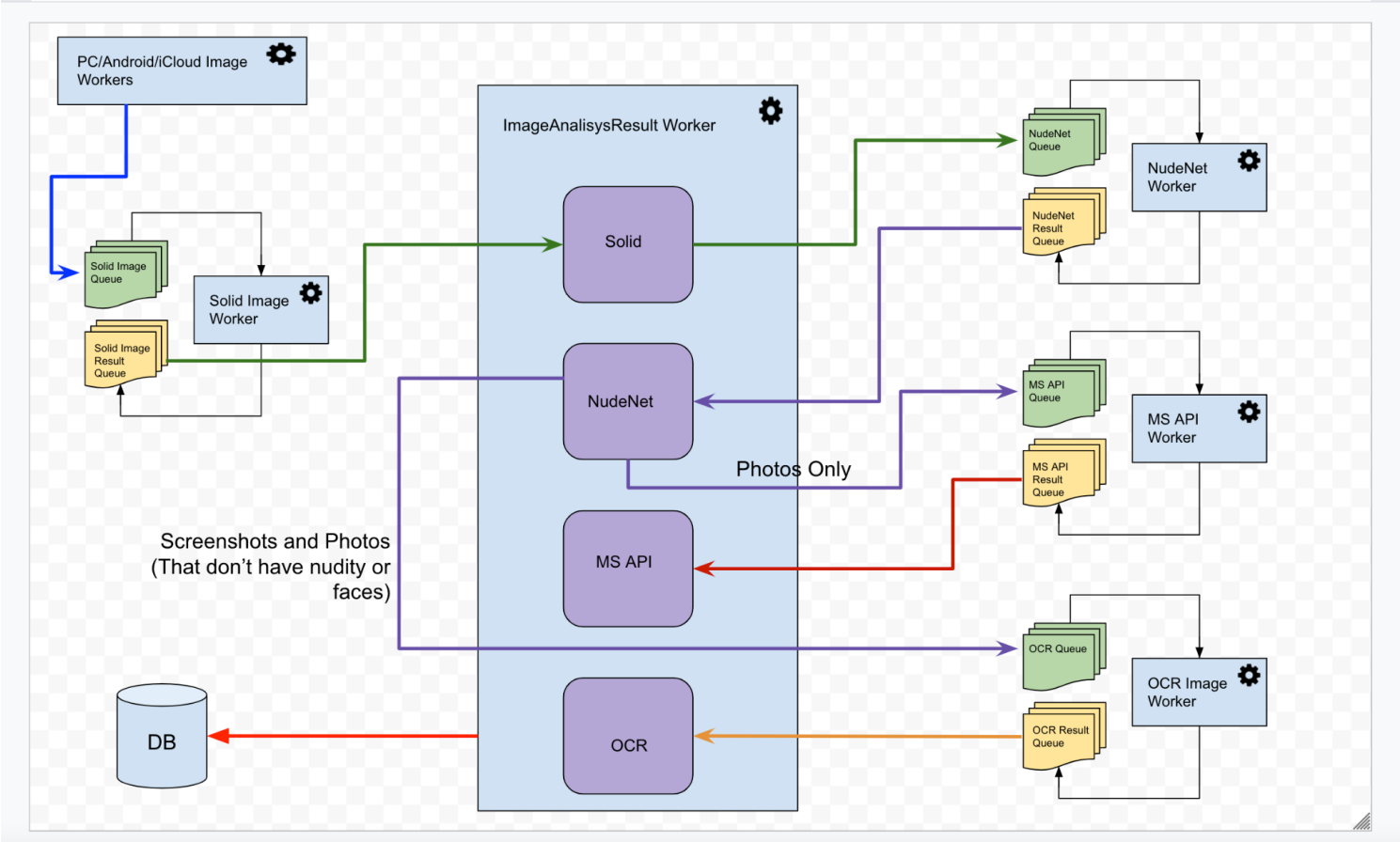
URL Validation Service

Open documentation task - ?

Image Analysis

Image analysis is a set of components to analyze images coming from user devices (PC/Mac/Android screenshots and iOS/Android user photos) to find adult content and recognize text on screenshots.

Schema



Solid Color

Ignored to reduce system load.

1. **Solid Image Analysis** algorithm should **take images from the queue** do the analysis and provide a result
    1. The result of processing the image through the Solid Image Analysis must be saved (to log or database) to be queried later if needed.
2. Based on the **result of the Solid Image** Analysis algorithm
    1. **If an image is Multicolor**, then it should be **put in a queue** to be analyzed by **NudeNet** algorithm
    2. **If an image is Solid Color**, then it should be **flagged as Low** severity. We need to Tag (create a TagId if needed) this image and indicate the **Cause "Solid color"**. This image should not continue processing.
    3. **If an image is too Small**, then it should be **flagged as Low** severity. We need to Tag (create a TagId if needed) this image and indicate the **Cause: "Image too small"**. This image should not continue processing.

NudeNet

NudeNet process based on Python nudenet library.

1. NudeNet algorithm should take the image from the corresponding queue do the analysis and provide a result
    1. The result of the process of the image through the NudeNet algorithm must be saved (to log or database) to be queried later if needed.
2. Based on the result of NudeNet algorithm
    1. If image labels include: *EXPOSED_GENITALIA_F, EXPOSED_GENITALIA_M, EXPOSED_BREAST_F, EXPOSED_BREAST_M, EXPOSED_ANUS, or EXPOSED_BUTTOCKS* with a Score greater or equal to 0.4, then the image should be flagged as High severity. We need to Tag (create a TagId if needed) this image and indicate the Cause “Erotic Content“. This image should not continue processing.
    2. If image labels include: *COVERED_BREAST_F, COVERED_GENITALIA_F, COVERED_BUTTOCKS, EXPOSED_BELLY or EXPOSED_ARMPITS* with a Score greater or equal to 0.4, then the image should be flagged as Medium severity. We need to Tag (create a TagId if needed) this image and indicate the Cause “Adult Content“. This image should not continue processing.
    3. If the event source is Android Photo, Android MMS or iOS Photo, and if the image labels include: *FACE_F or FACE_M* with a Score greater or equal to 0.25, then the image should be added to the queue to be processed for MS API analysis.
    4. If the event source is Android Photo, Android MMS or iOS Photo, and the image labels include: *FACE_F or FACE_M* with a Score lower than 0.25 OR does not return the *FACE_F or FACE_M* labels, then the image should be added to the queue to be processed by OCR algorithm.
    5. If the event source is Android Screenshot, or Mac Screenshot or Windows Screenshot, and if the image did not match on any of the points a. or b. then the image should be added to the queue to be processed by OCR algorithm.

OCR

We are using the IconOCR library to recognize image text.

1. OCR algorithm should take the image from the corresponding queue do the analysis and provide a result
    1. The result of the OCR process must be saved (to log or database) to be queried later if needed. We only need to save the full result, not the individual records.
2. With the result of OCR algorithm
    1. If the image contains words, those words needs to be analyzed using the rules that correspond with the monitoring profile and text-only rules.
    2. If any rule matches, then the image should be flagged according to the severity associated to the rule.
    3. If no rules match, then the image should be flagged as Low severity.

We are using Microsoft Cognitive Web Services to analyze images.

The MS API algorithm should take the image from the corresponding queue, perform the analysis, and provide a result.

1. The MS API should include a new step before invoking the MS API that checks if the file size is greater than 4MB, in which case it will produce a resized image to be sent to MS API. Once the image is processed the resized version should be deleted.
2. Based on the result, the MS API should flag the image with the corresponding severity. High, Medium or Low. The image should not continue processing.

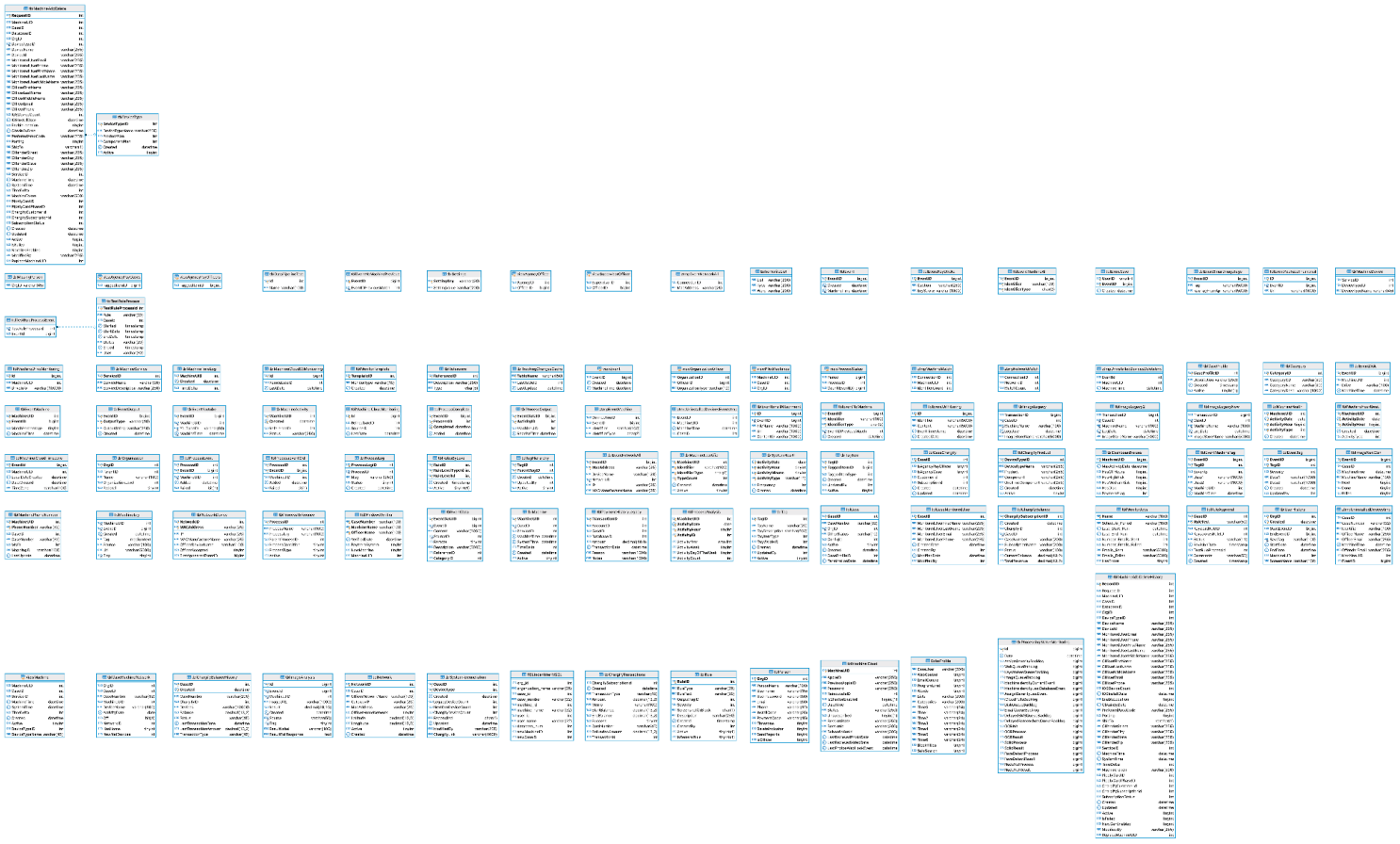
DevOps Notes

Git

We are using AWS CodeCommit.
Clone URL: https://git-codecommit.us-east-2.amazonaws.com/v1/repos/NextGen

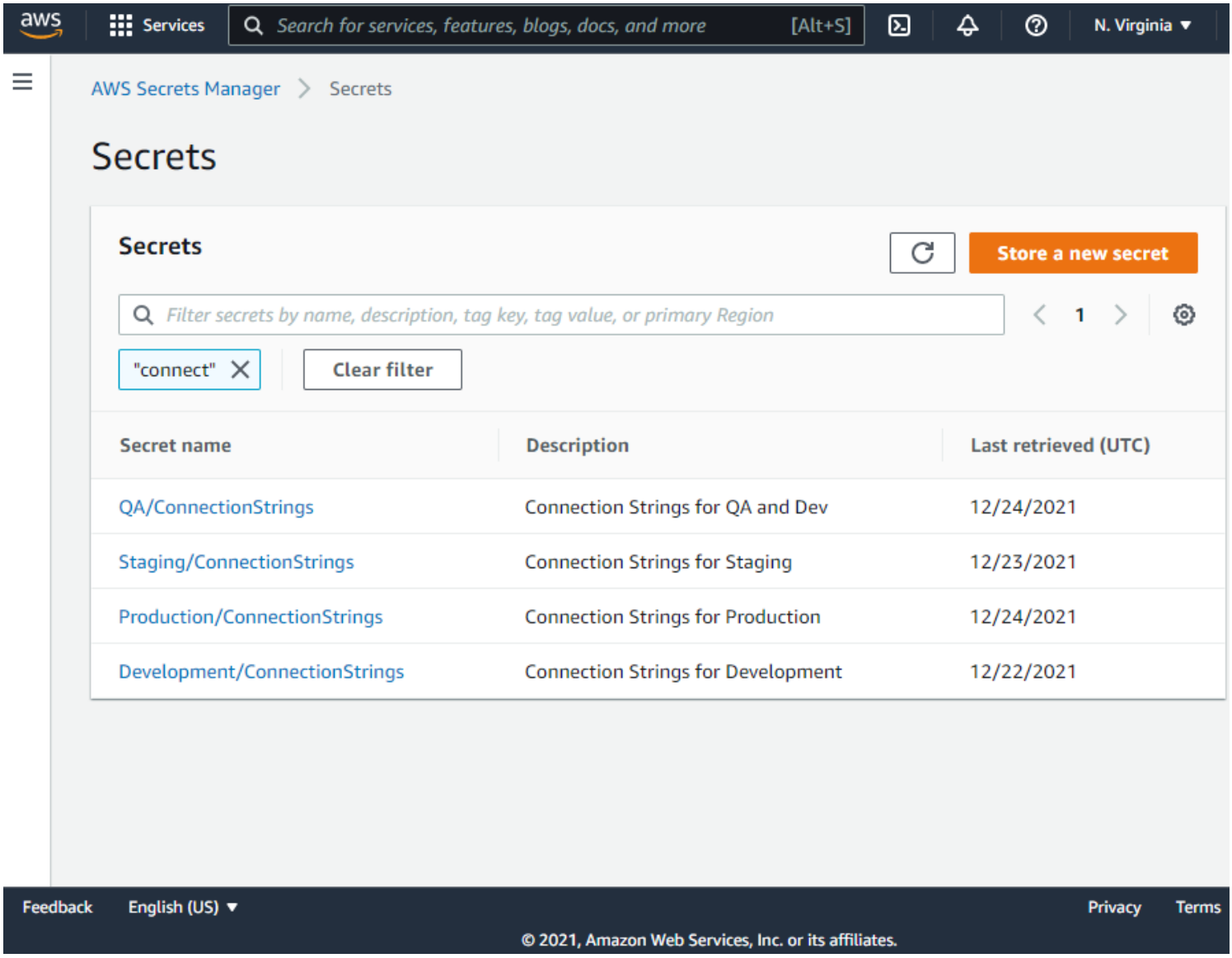
Database Schema

(Note - this can be provided as a PNG file upon request).



Accessing databases

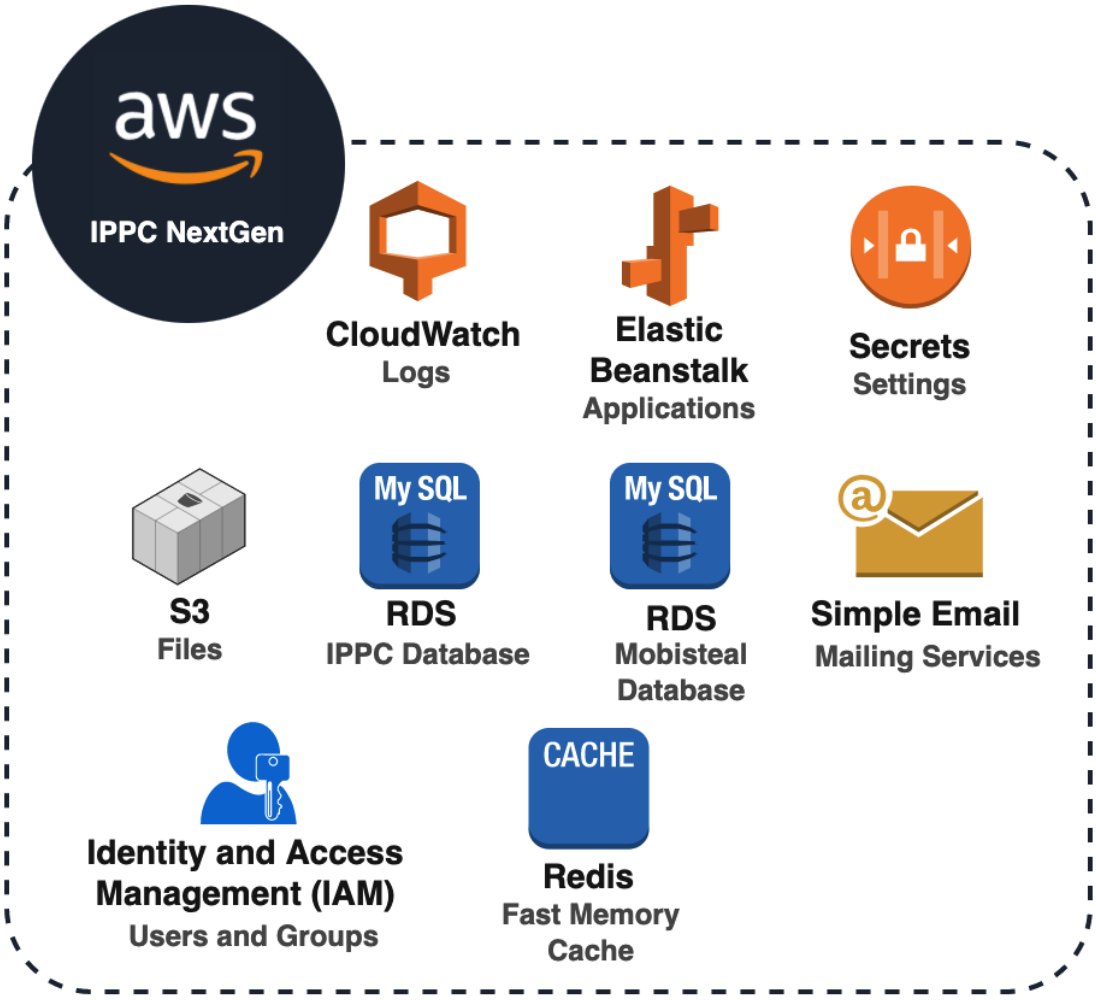
Connection strings are available in AWS Secrets Service as [Environment]/ConnectionStrings secrets (separate secret for each environment).



OwnDbConnection - NextGen MySQL database
AndroidDbConnection - Mobi MySQL database (Android data)
LegacyConnectionString - Legacy MS SQL Server database

You will need to make sure that access is open for your IP in AWS Security Groups or connect via VPN.

AWS


aws
IPPC NextGen
CloudWatch
Logs
Elastic
Beanstalk
Applications
Secrets
Settings
S3
Files
My SQL
RDS
IPPC Database
My SQL
RDS
Mobisteal
Database
Simple Email
Mailing Services
Identity and Access
Management (IAM)
Users and Groups
CACHE
Redis
Fast Memory
Cache

AWS Resources

Below the list of Elastic Beanstalk, Elastic Cache (redis), RDS, DynamoDB resources.

This list can be provided as an Excel file, Google Sheet or PDF.
AWS Resources

AWS S3

The NextGen application components use S3 to store files.
Full list of S3 buckets you can find in following documents AWS Resources

PC Event Processing (including legacy preprocessor and next gen worker) using s3 to store uuencoded files, screenshots and email attachments

Buckets:
```
ippc-bucket-preproccessor-dev
ippc-bucket-preproccessor-prod
Ippc-bucket-preproccessor-qa
```

Reicubate Service and iCloud Next Gen worker components using s3 to store icloud photos/vidoes, description files and processing photo/video results.

Buckets:
```
ippc-reincubate-data
ippc-reincubate-data-prod
ippc-reincubate-data-processing
ippc-reincubate-data-processing-prod
ippc-reincubate-data-processing-done
ippc-reincubate-data-processing-done-prod
```

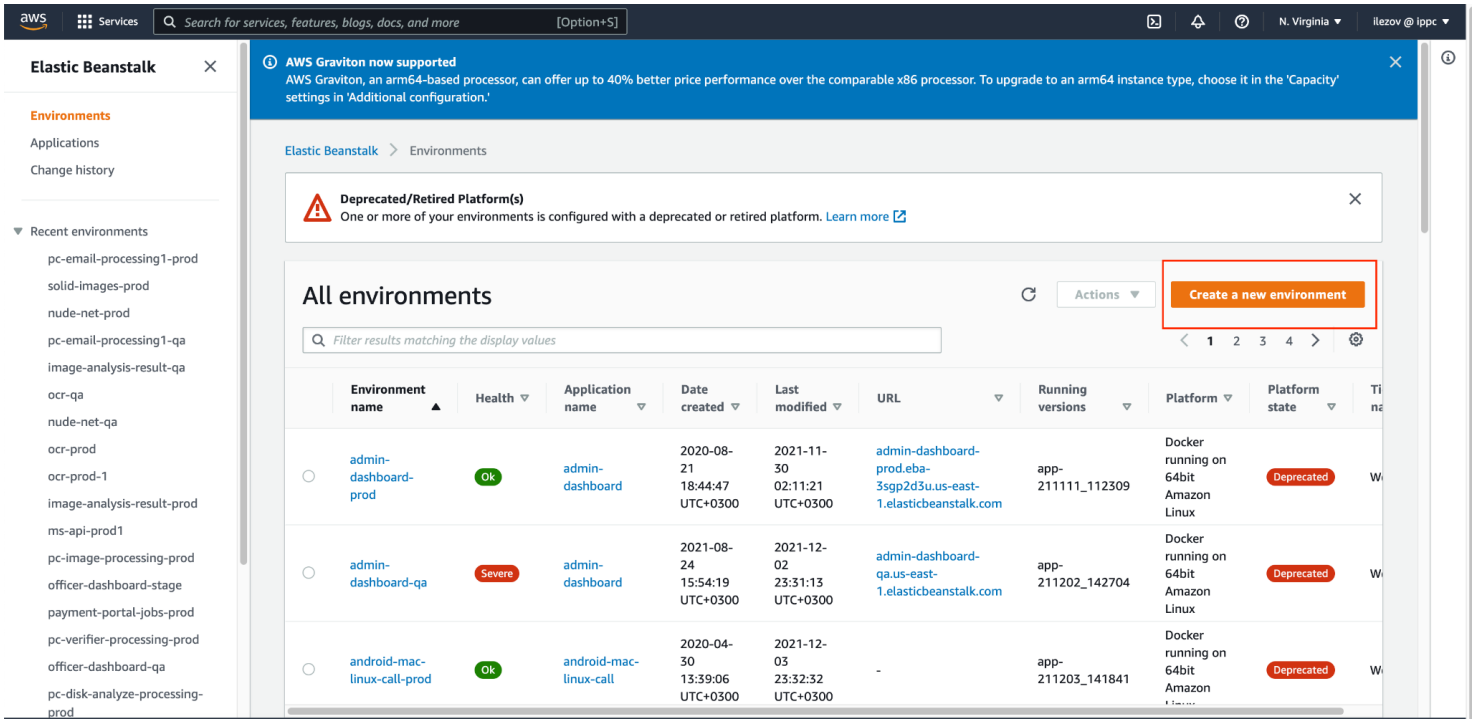
Also we store config for web search processing worker in s3

Buckets:
```
ippc-bucket-config-files
```

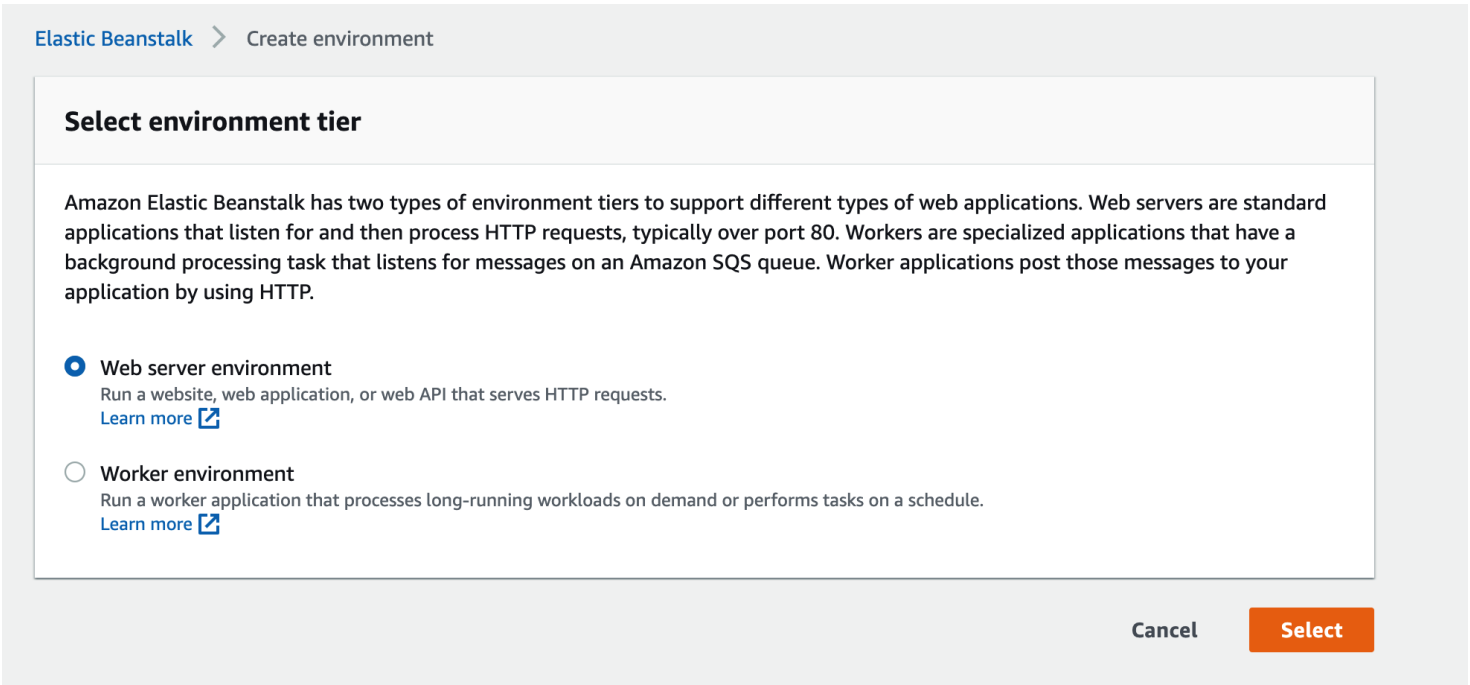
AWS Elastic Beanstalk

How to create a new EB environment

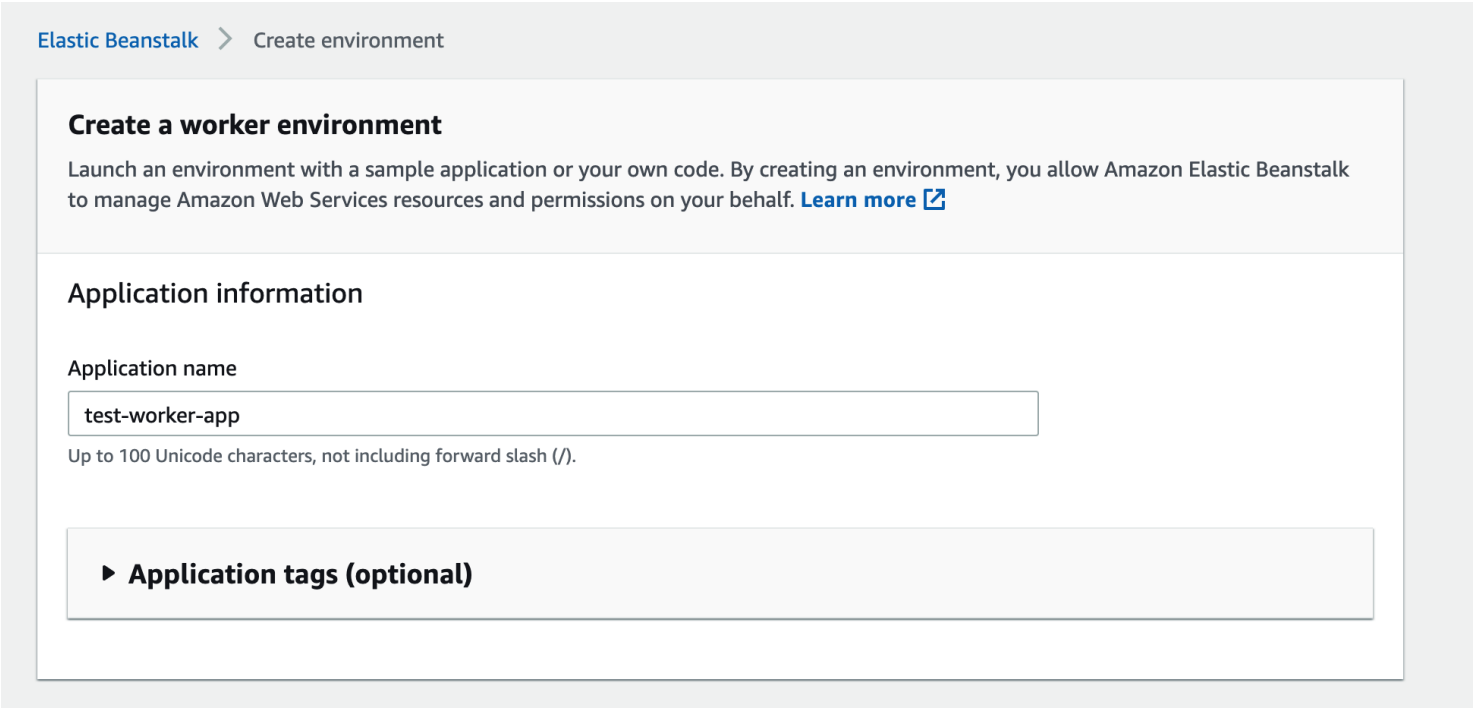
On EB environments list page click to “CREATE NEW ENVIRONMENT”



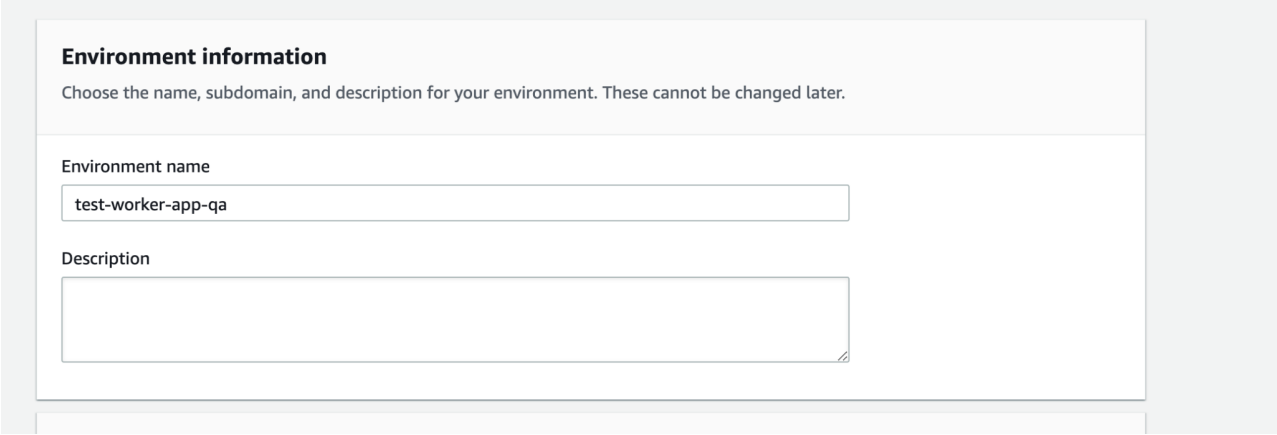
Select environment type “Web App” or “Worker App” and click “SELECT”



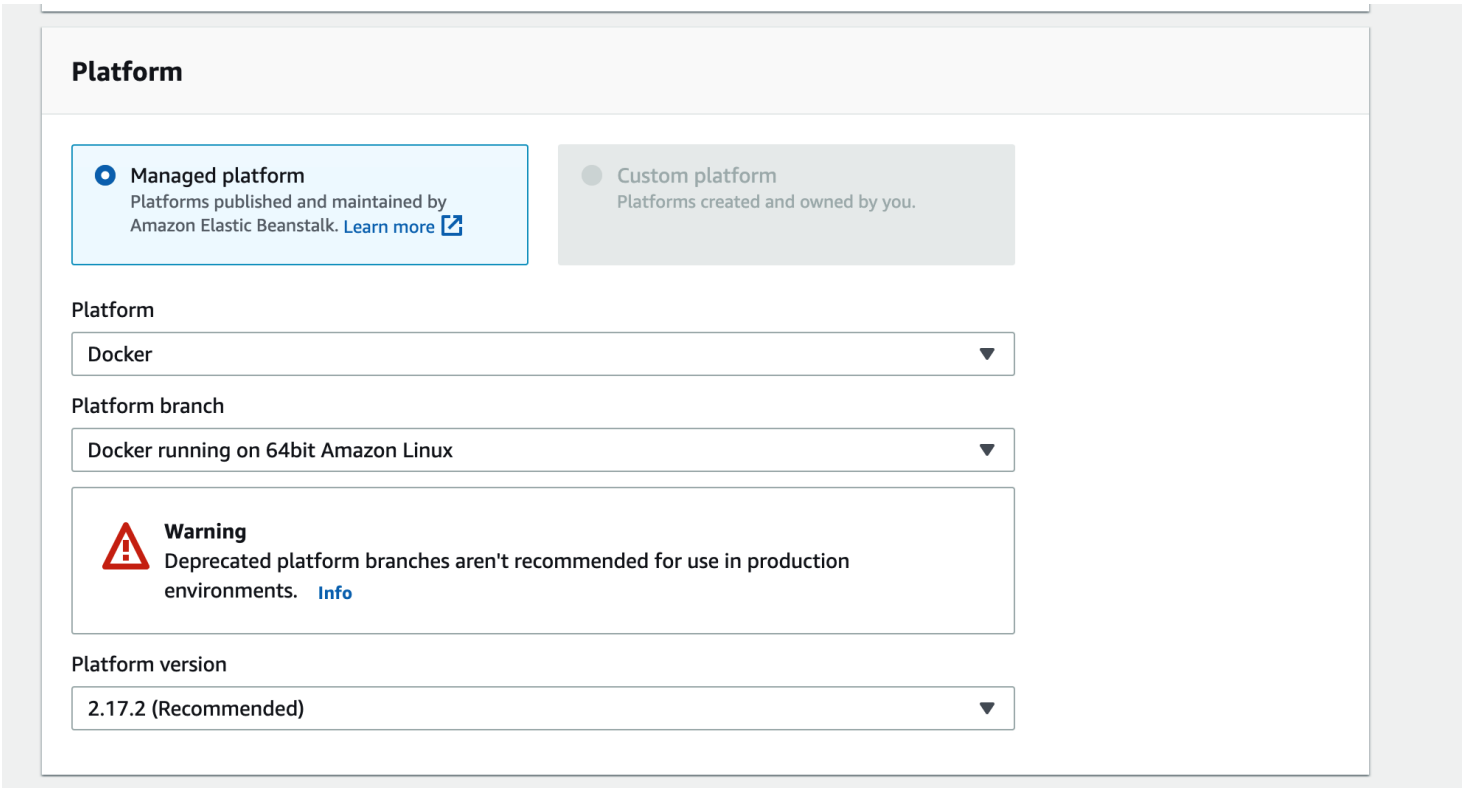
Enter application name



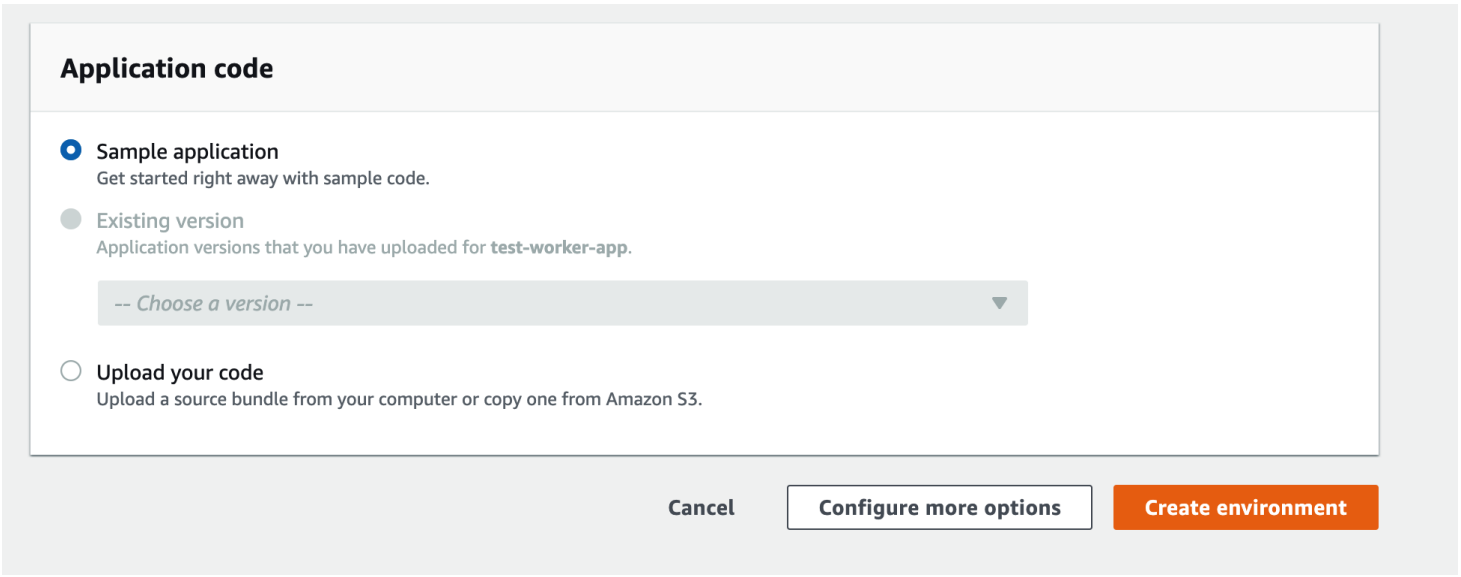
Enter environment name (name convention is application name and qa or prod prefix)



Select the platform it should be “Docker” for both web and worker application.
Platform branch should be “Docker running on 64bit Amazon Linux”



Click to “CREATE ENVIRONMENT”



How to deploy an existing application to EB

For example - deploy Officer Dashboard application to EB

1. Go to project folder `\NextGen\processing\Processing`
2. Run .bat deployment script for application what you want to deploy (deploy-QA-officer-dashboard.bat or deploy-PROD-officer-dashboard.bat)
3. Wait when deployment finished

Create a deployment script for a new environment

1. For example - we created new EB environment **test-app-qa**
2. Also we create a new web project `TestApp.Web`
3. We need to create an EB config in the project folder. Go to `\NextGen\processing\Processing\TestApp.Web` and run `eb init` in console (aws cli install required)
4. Select region (**us-east-1**), select application (**test-app**), select environment (**test-app-qa**)
5. Create new Dockerfile in `\NextGen\processing\Processing\TestApp` folder with content

```
#See https://aka.ms/containerfastmode to understand how Visual Studio
uses this Dockerfile to build your images for faster debugging.
FROM mcr.microsoft.com/dotnet/core/aspnet:latest AS base
WORKDIR /app
EXPOSE 80

FROM mcr.microsoft.com/dotnet/core/sdk:latest AS build
WORKDIR /app
COPY . .
RUN dotnet restore "TestApp.Web/TestApp.Web.csproj"
RUN dotnet publish "TestApp.Web/TestApp.Web.csproj" -c Release -o out

FROM base AS final
WORKDIR /app
COPY --from=build /app/out .
ENTRYPOINT ["dotnet", "TestApp.Web.dll"]
```

6. Create `deploy-QA-test-app.bat` file in `\NextGen\processing\Processing` folder with content

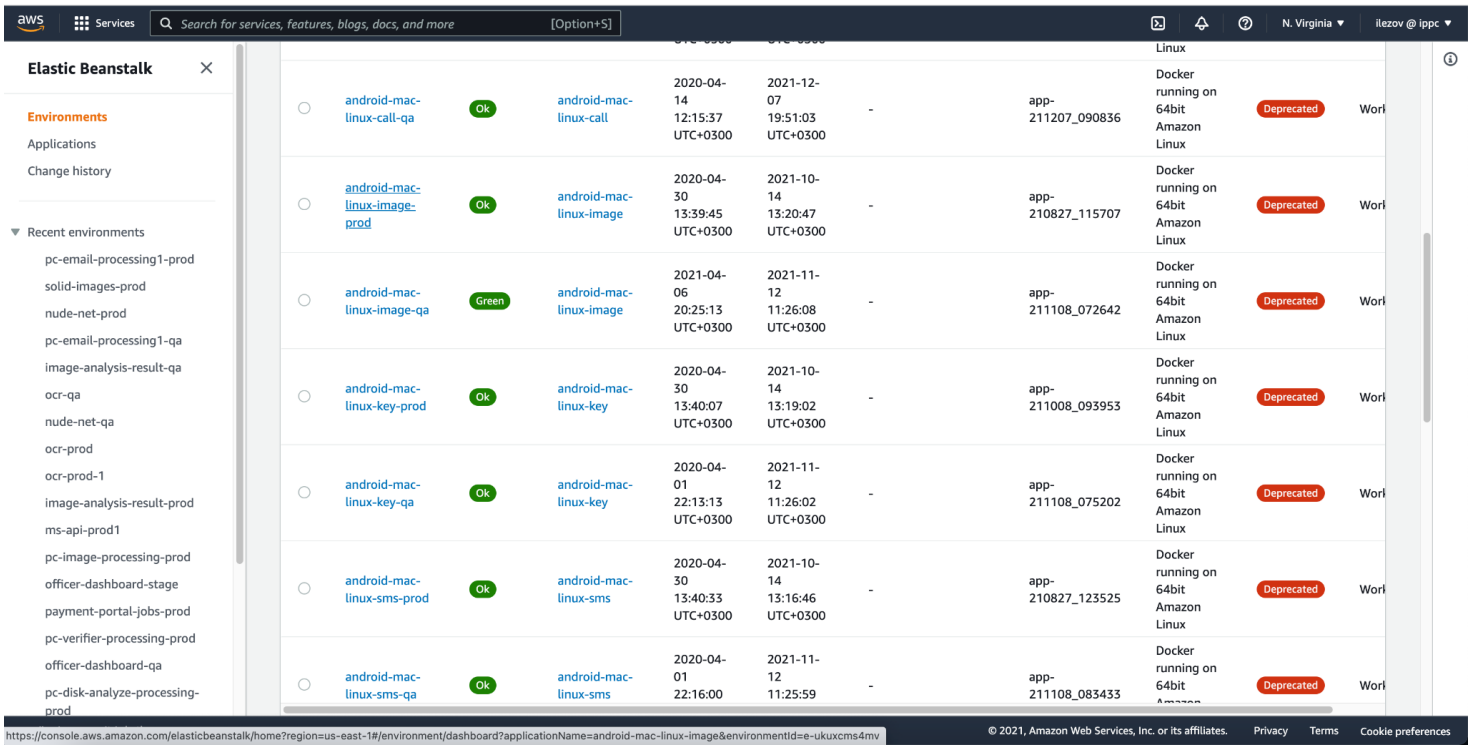
```
del Dockerfile
del .ebignore
rd /s /q ".\.elasticbeanstalk"
xcopy ".\TestApp.Web\.elasticbeanstalk" ".\.elasticbeanstalk" /e /i /h
COPY ".\TestApp.Web\Dockerfile" ".\Dockerfile"
COPY ".\TestApp.Web\.ebignore" ".\.ebignore"
eb deploy test-app-qa --region us-east-1
del Dockerfile
```

```
del .ebignore
rd /s /q ".\.elasticbeanstalk"
Pause
```

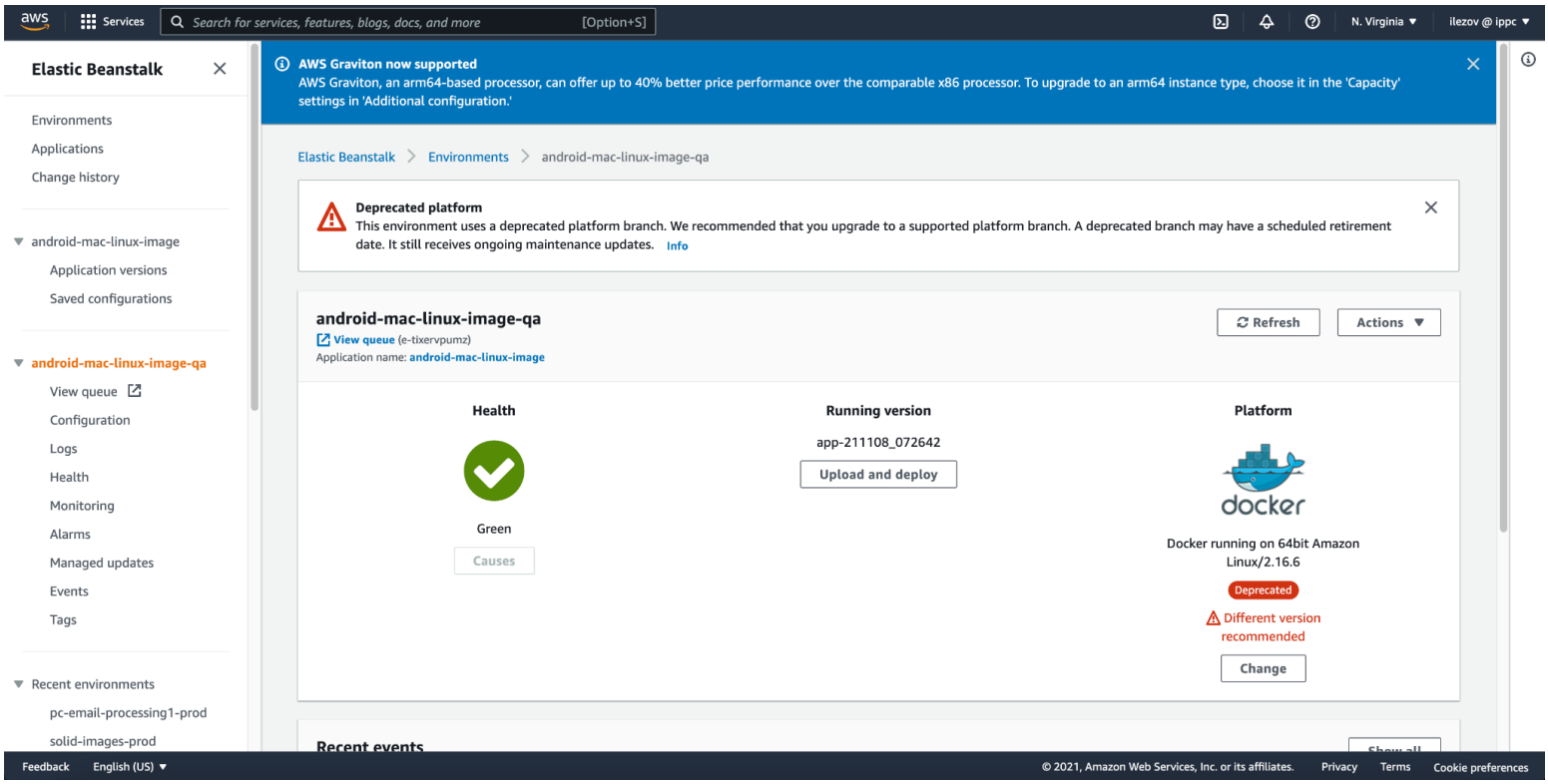
7. Run `deploy-QA-test-app.bat`

Scale up/down and stop an EB environment

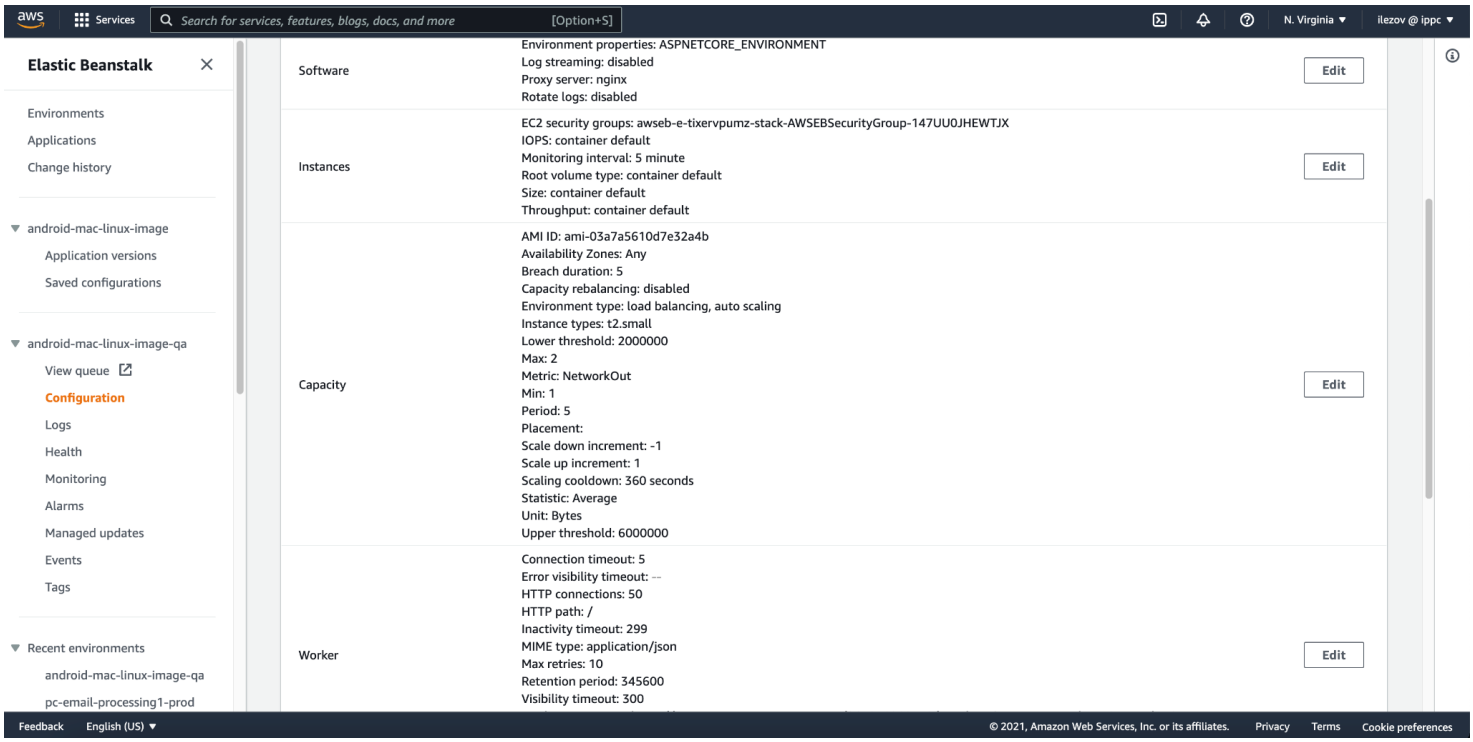
Go to EB environment list and click to “environment”



Go to environment “Configuration”

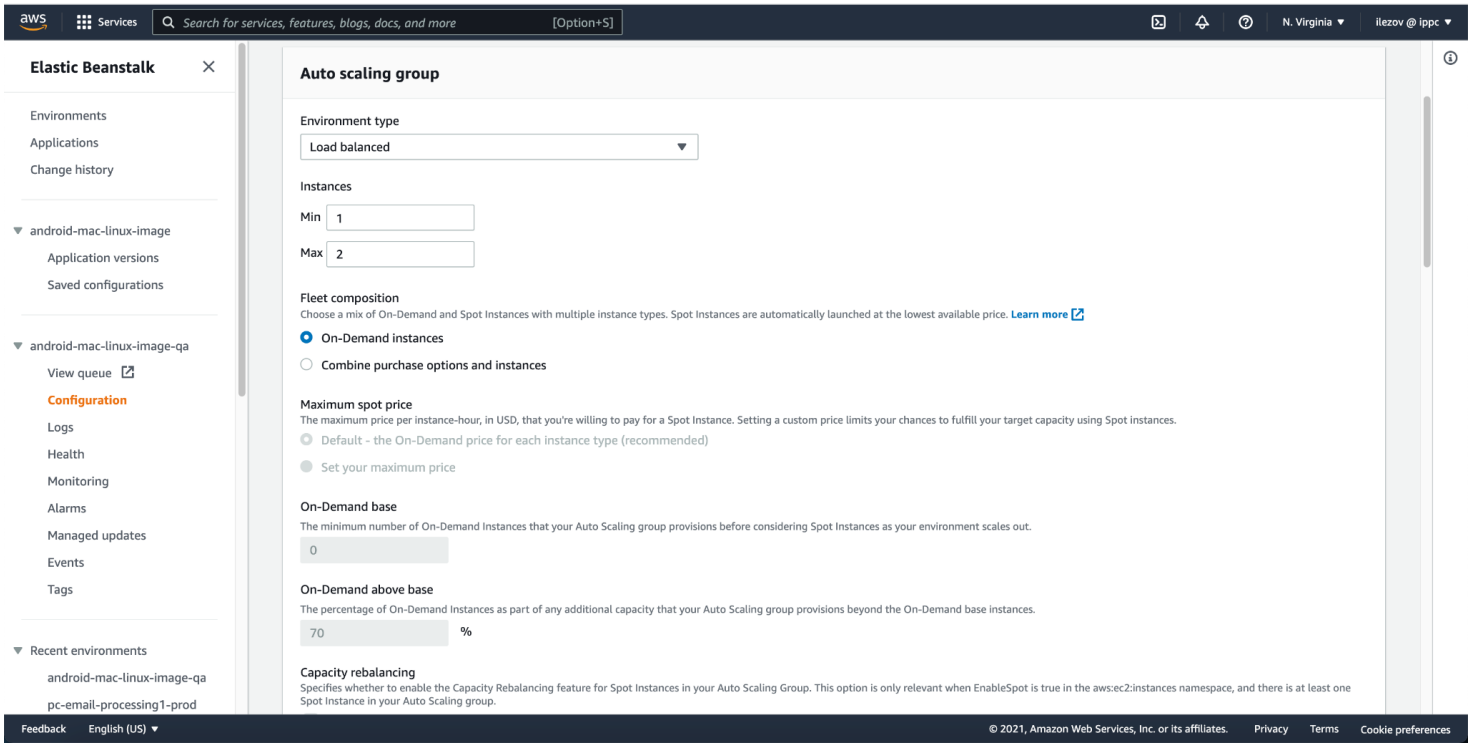


Click to “EDIT” for “Capacity” option group



Change “Instances” “Min” and “Max” values

1. Increase to scale up
2. Decrease to scale down
3. Set both 0 to stop

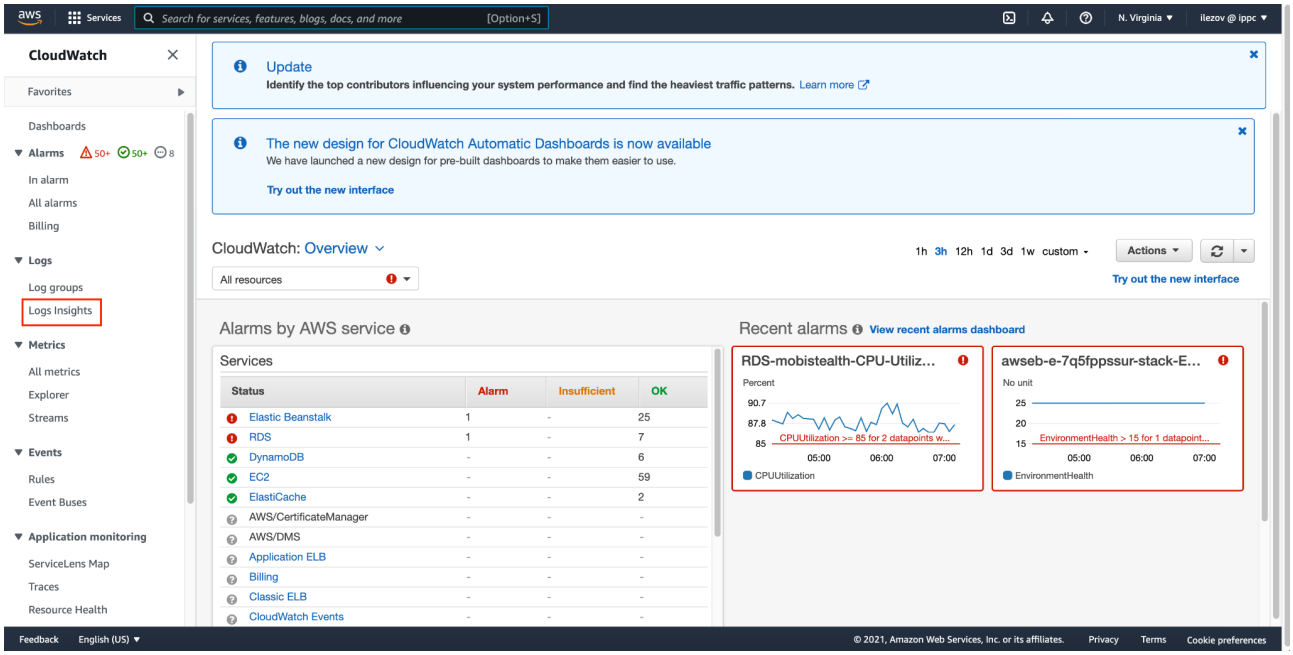


CloudWatch and application logs

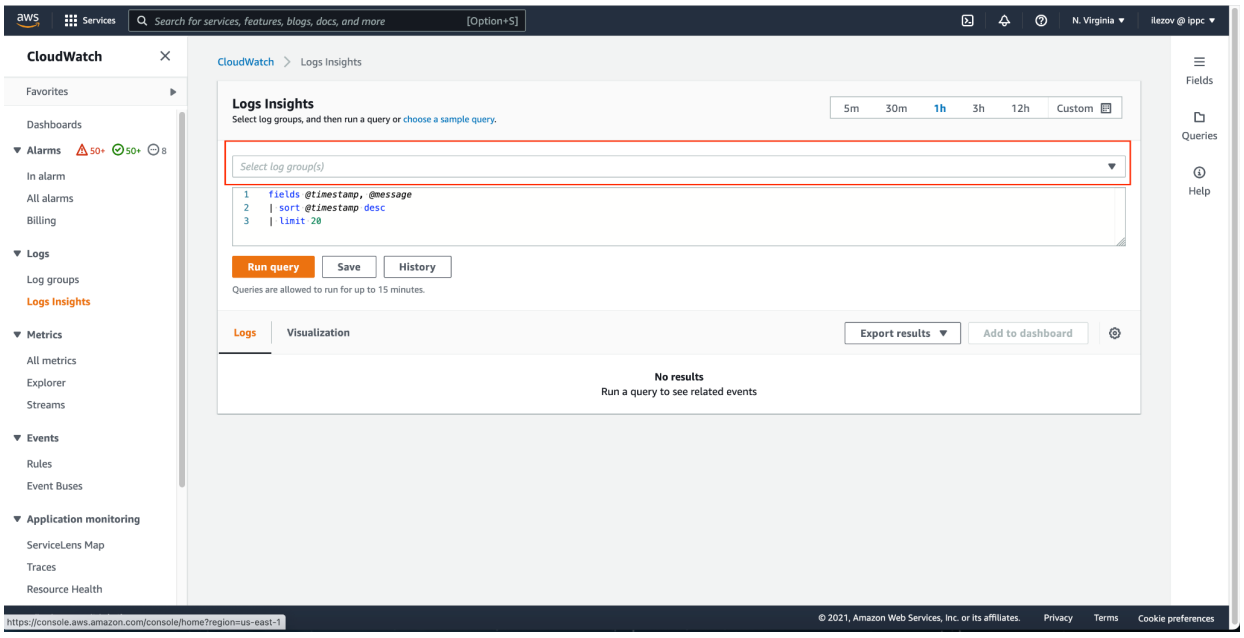
Each application writes logs to some log group in AWS Clloudwatch.

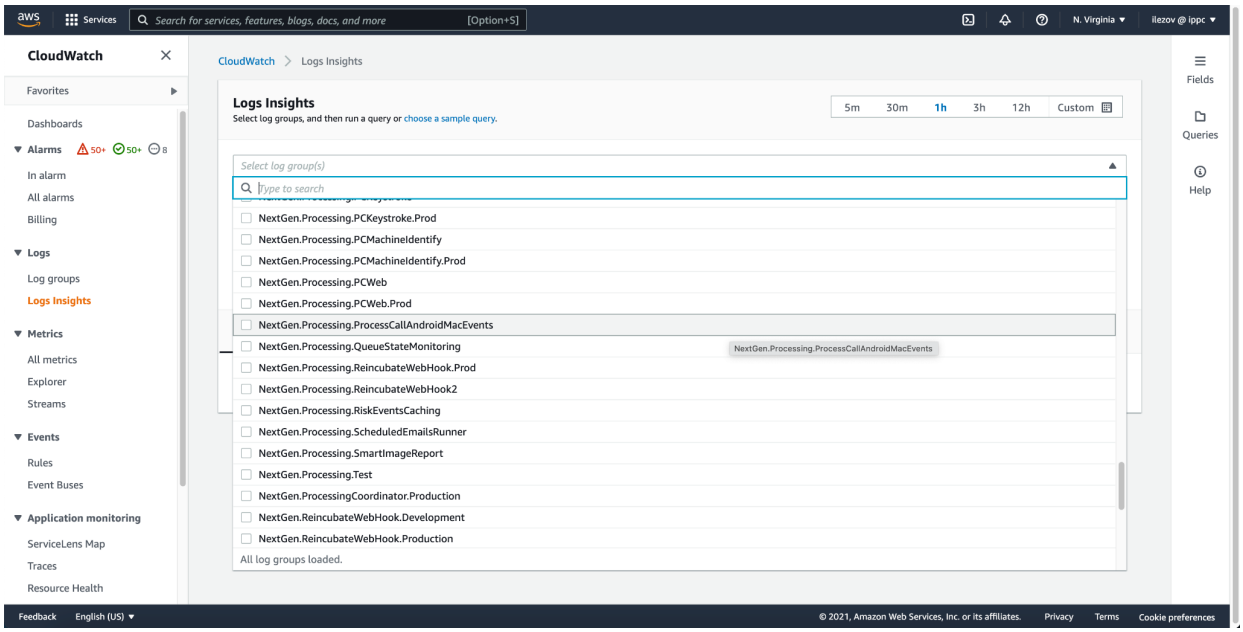
Here are steps on how to get logs for a group.

Go to the Cloudwatch page and click to Logs Insights

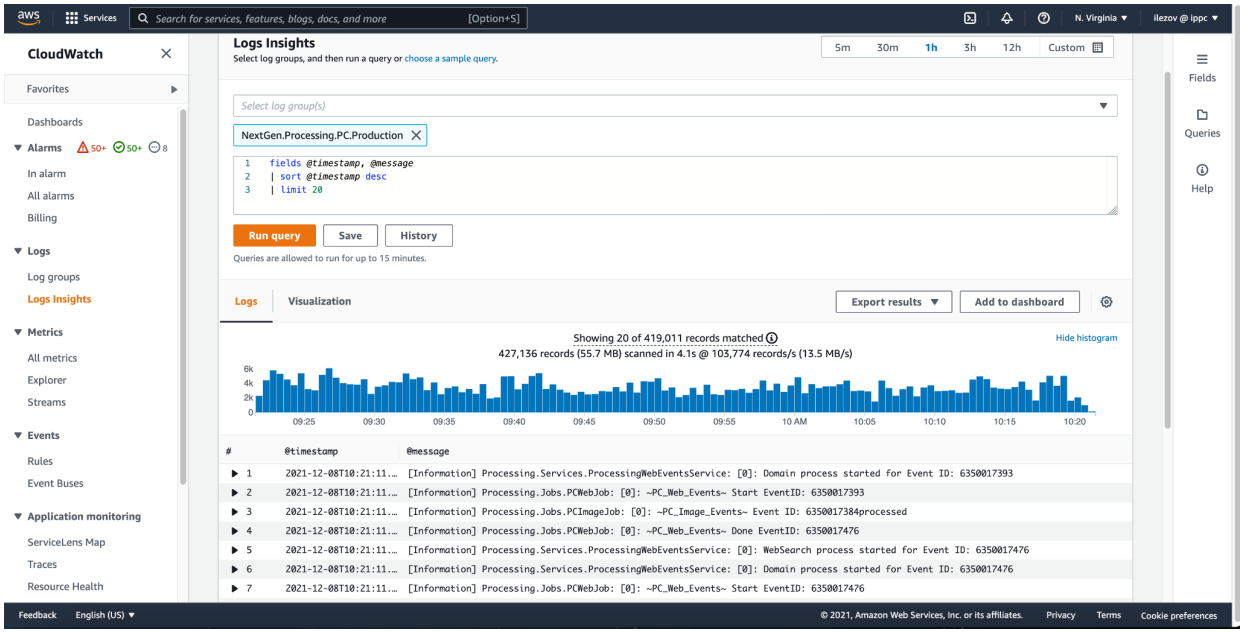


Select log group

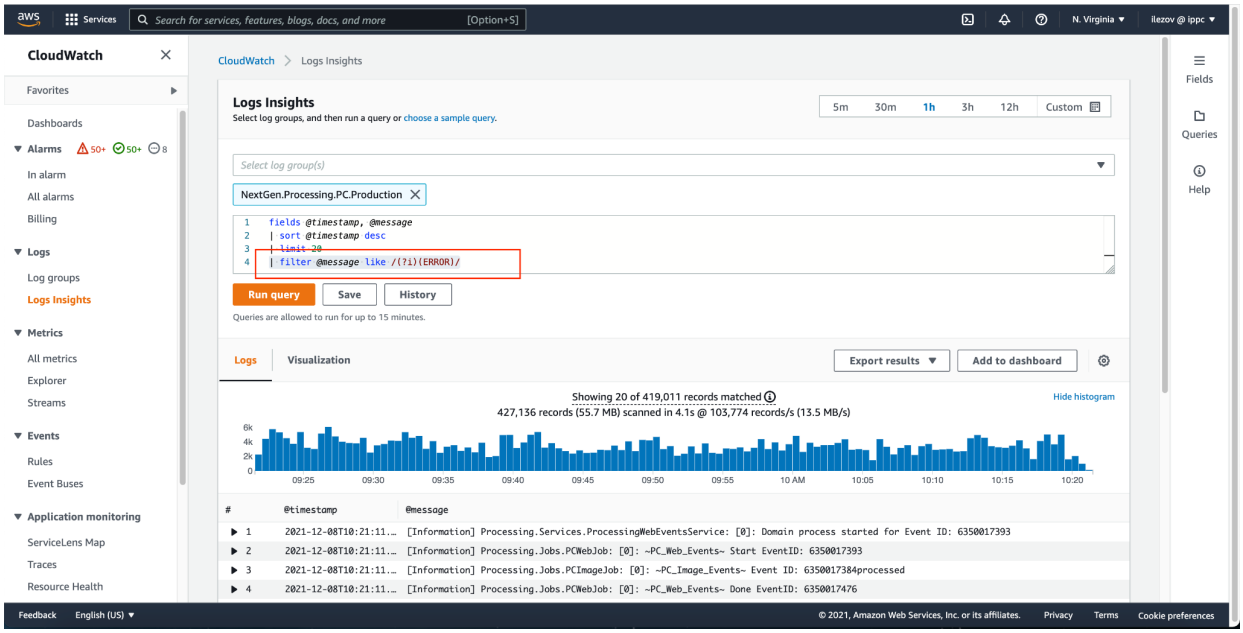




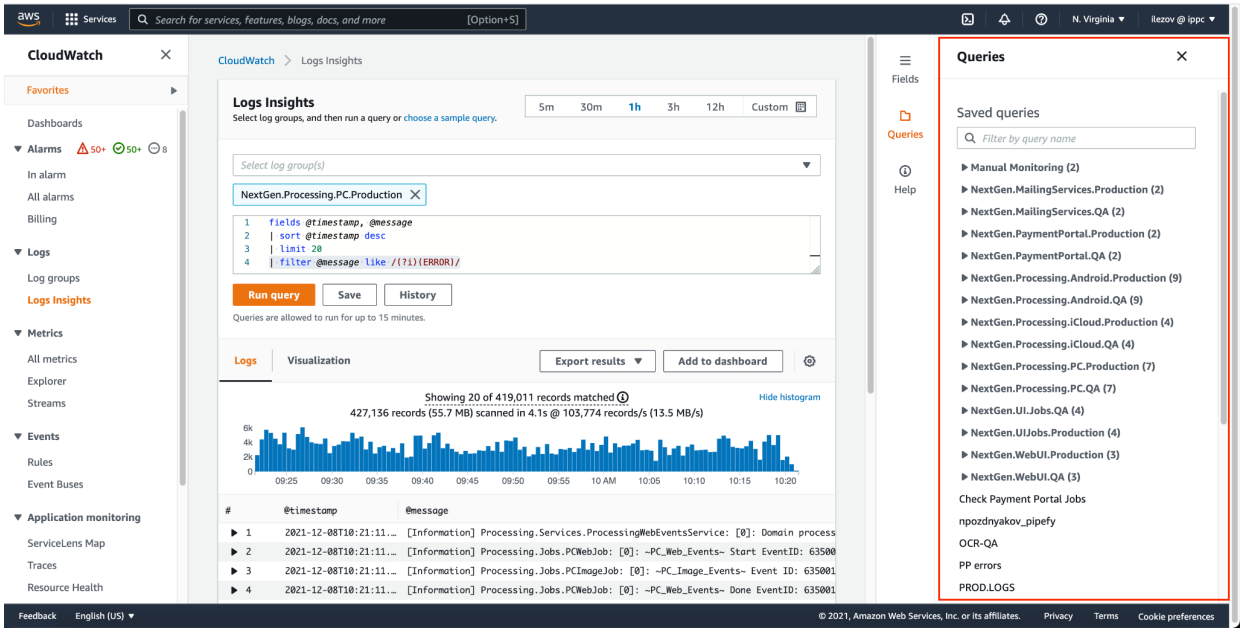
And click run query



You can change the time range and filter results.
To filter results add a “filter” condition to the query.



You can also use presaved queries



Reprocessing devices

Android & MacOS devices

Note - Currently there is no way to reprocess a time range, the workaround is to reprocess device for all time

1. Remove all data from device in tblEventMachine

```
DELETE FROM tblEventMachine WHERE MachineUID=[MUID]
```

2. Remove all data from device in cache table tblEventMachineTag

```
DELETE FROM tblEventMachineTag WHERE MachineUID=[MUID]
```

3. Run reprocess tool on “ippc-domain” server for each event type

```
cd C:\Avk\android\log_cal_calls_1\Debug
ProcessAndroidMacEvents.exe [MUID]

cd C:\Avk\android\log_act_activity_1\Debug
ProcessAndroidMacEvents.exe [MUID]

cd C:\Avk\android\log_pct_chat_1\Debug
ProcessAndroidMacEvents.exe [MUID]

cd C:\Avk\android\log_pic_picture_1\Debug
ProcessAndroidMacEvents.exe [MUID]

cd C:\Avk\android\log_sms_text_message_1\Debug
ProcessAndroidMacEvents.exe [MUID]

cd C:\Avk\android\log_vdo_video_1\Debug
ProcessAndroidMacEvents.exe [MUID]

cd C:\Avk\android\log_web_history_1\Debug
ProcessAndroidMacEvents.exe [MUID]
```

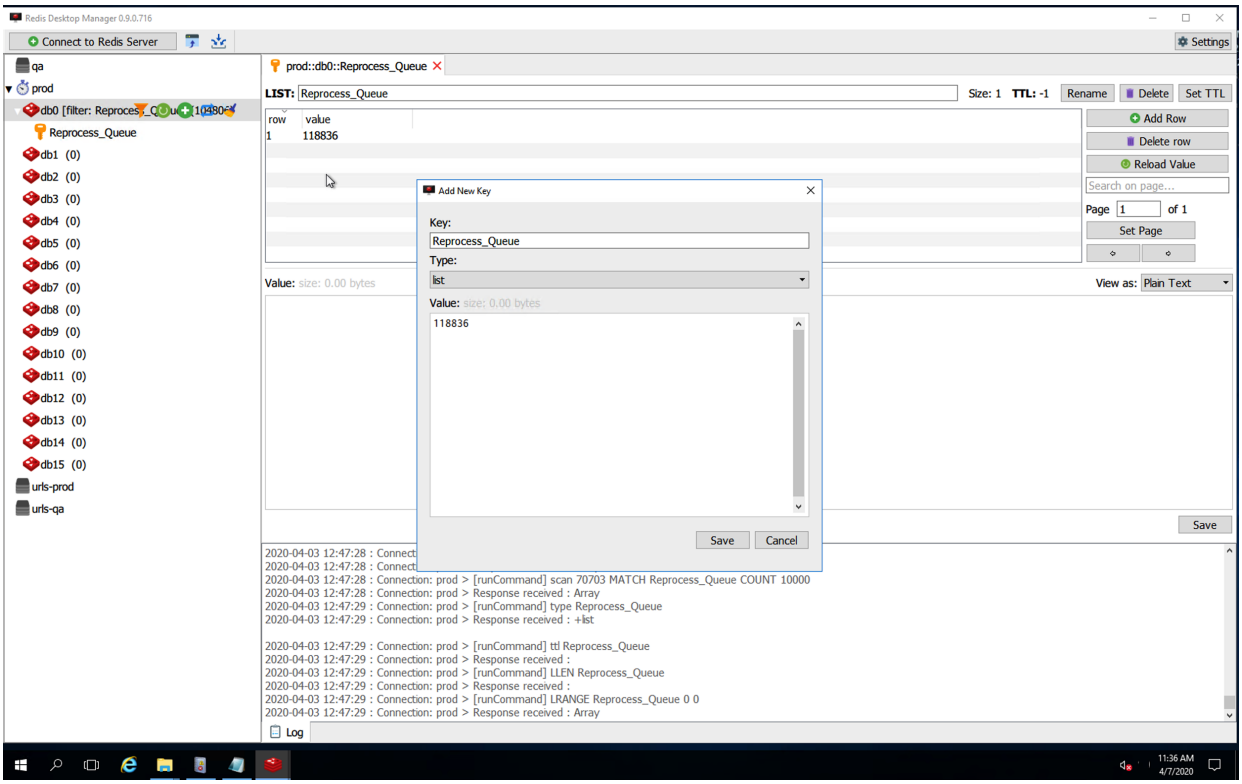
4. Refill data in cache table

```
INSERT IGNORE INTO tblEventMachineTag (EventID, TagID, Severity, Data1, Data2,
MachineUID, MachineTime)
select et.EventID, et.TagID, et.Severity, et.Data1, et.Data2, e.MachineUID as MachineUID,
e.MachineTime
FROM tblEventTag et
left JOIN tblEventMachine e ON et.EventID = e.EventID
WHERE e.MachineUID=[MUID] and et.Severity in (5,10);
```

PC devices

Reprocess tool for PC device reprocess result for last 90 days

1. Go to "ippc-domain" server
2. Create in prod redis store **Reprocess_Queue** key with [MUID] values with a type "list"



3. Run `C:\Avk\pc\Debug\ReprocessTool.exe`

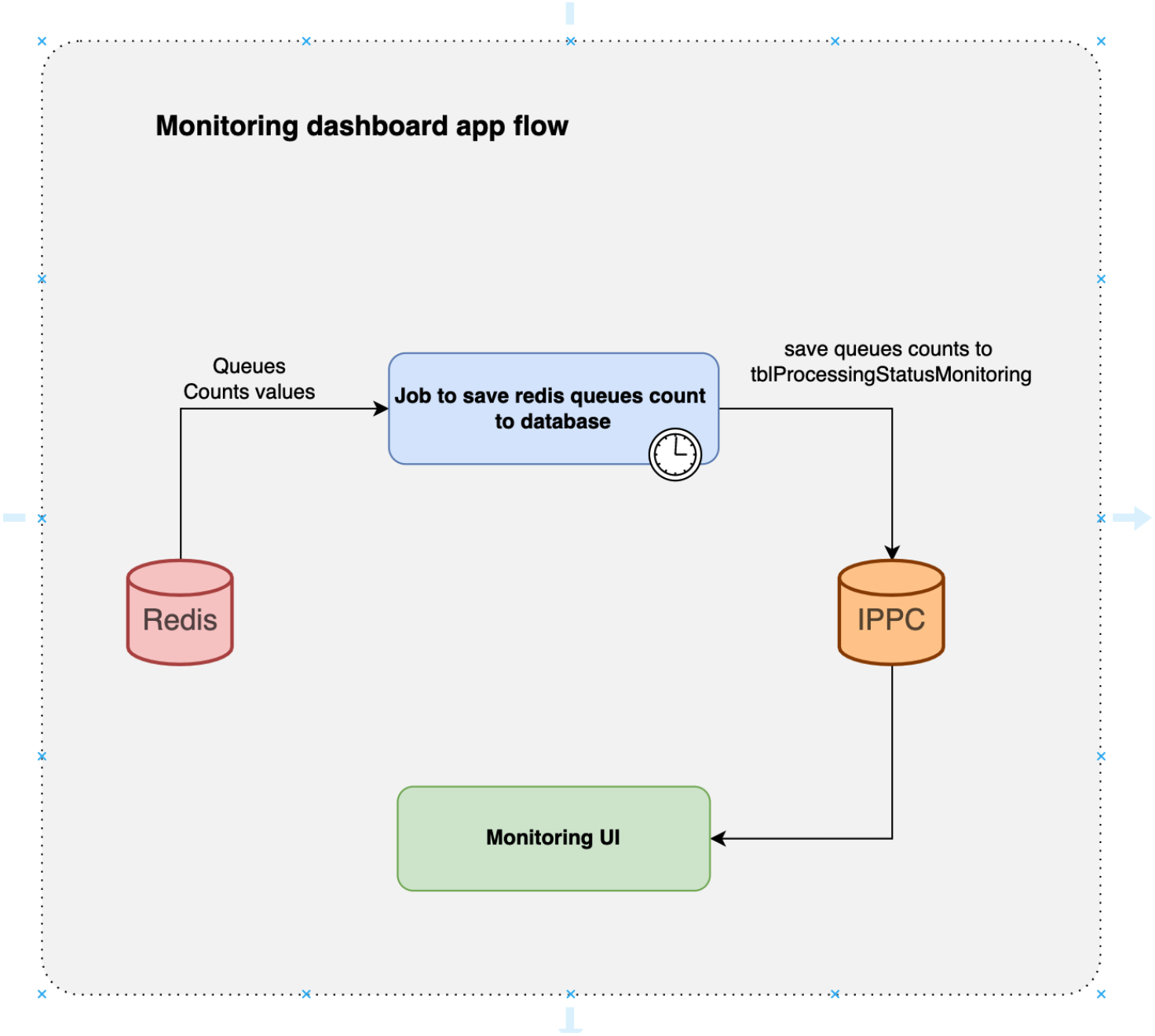
iOS devices

Requires support from the Reincubate team to ask to send iCloud files again.

Monitoring Dashboard

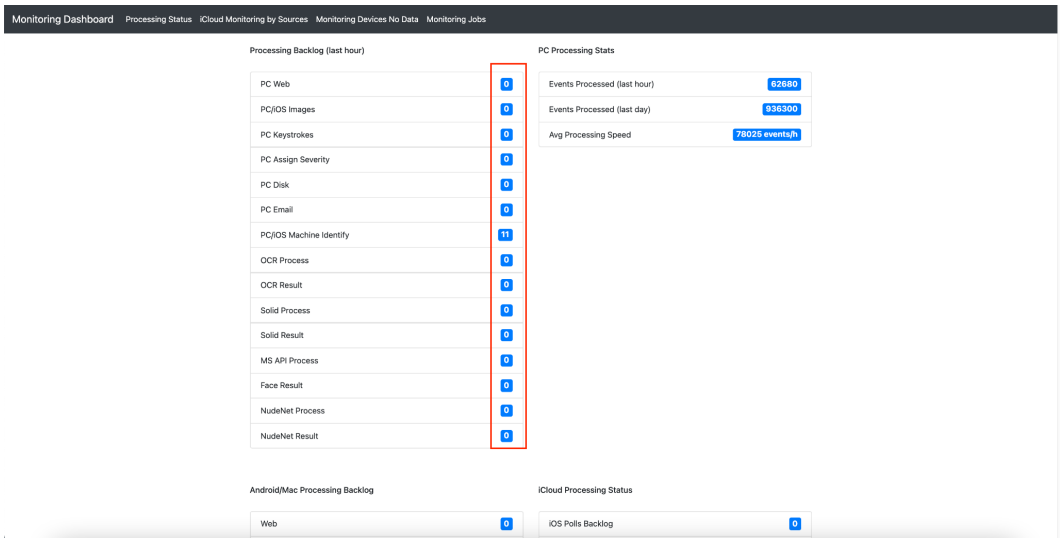
Note - Some processing parts can run slow for unknown reasons and jobs can fail.

To monitor current state of processing we have a monitoring dashboard.



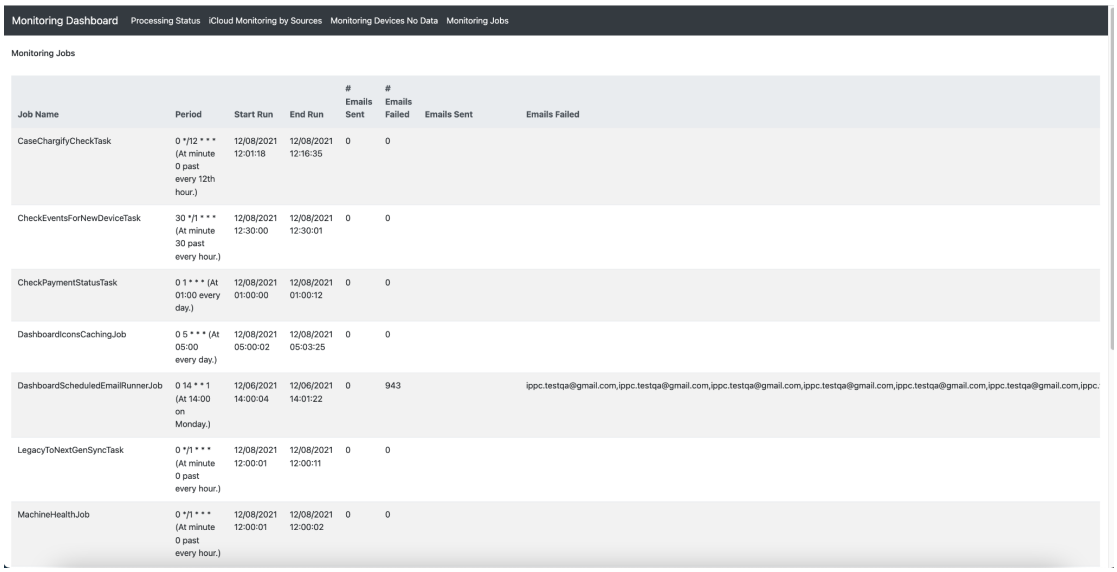
Backlog

To monitor events processing backlog - go to the home page
All values should be < 100000



Jobs

We can monitor job run status on monitor jobs page

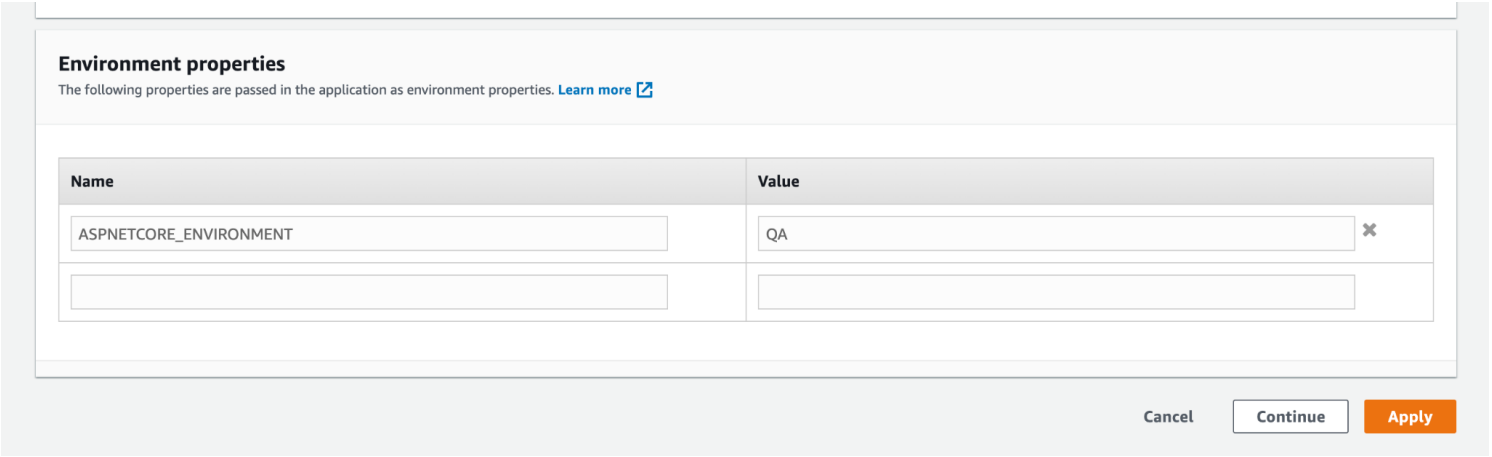


Secrets/Configuration

We are using AWS Secrets to store application configuration and get secrets in code by AWS API key.
Secrets on AWS are grouped as follows. These are a settings for Production, the same settings are available for QA and Staging:

Manage application configuration

EB environment > configuration > Software > Edit



ASPNETCORE_ENVIRONMENT variable manage which configuration from secrets will be applied to application “Staging” value for staging environment, “Production” for production, “QA” for qa.

Deployment downtime handling & process

Processing

Processing deployment doesn’t introduce any noticeable downtime for users, so it can be deployed any time if approved by IPPC.

Potential issue with processing downtime is the growing event backlog (not processed data). The longer processing is stopped the more it takes to catch up (the more it takes for Officers to see their data). It is not recommended by IPPC to stop Legacy processing for more than 8 hours.

Typically deployment is quick and doesn’t increase backlog significantly. As opposed to deployment, database maintenance may take hours and if you need to stop processing for that you should keep in mind the growing backlog.

You may think that catching up will take nearly the same amount of time that processing was down (the time can be reduced by scaling processing up).

UI

UI downtime (Officer Dashboard) impacts users and should be planned for off hours (US Eastern) and is always coordinated with IPPC.

Deployment dependencies

All system components (processing workers, UI apps, scheduled job instances) can be deployed separately and do not depend between each other
But there are code dependencies and one change in code can be affected to deploy more than one component.
It require some manual tracking after code changes and what deployment components affected by these changes.

Configured alerts & expected monitoring plans (health checks)

Cloudwatch Alerts

AWS Service: Beanstalk Environments		
Alert Name	Condition	SNS Topic/Action
-EnvironmentHealth-	EnvironmentHealth > 10 for 1 datapoints within 15 minutes	BeanstalkAlerts
-CPUUtilization-	CPUUtilization >= 90 for 1 datapoints within 5 minutes	BeanstalkAlerts
-NetworkOut-	NetworkOut <= 0 for 1 datapoints within 30 minutes	BeanstalkAlerts
AWS Service: RDS		
awsrds-ippc-prod-High-CPU-Utilization	CPUUtilization >= 50 for 1 datapoints within 5 minutes	RDSNotifications
awsrds-ippc-prod-Low-DB-Connections	DatabaseConnections <= 5 for 1 datapoints within 5 minutes	RDSNotifications
awsrds-ippc-prod-Low-Free-Storage-Space	FreeStorageSpace <= 52428800000 for 1 datapoints within 5 minutes	RDSNotifications
awsrds-ippc-prod-Low-Network-Transmit-Throughput	NetworkTransmitThroughput <= 0 for 1 datapoints within 5 minutes	RDSNotifications
AWS Service: ElasticCache		
ElasticCache-PROD-OutOfMemory	DatabaseMemoryUsagePercentage > 90 for 2 datapoints within 10 minutes	IPPC-and-EPIC
ElasticCache-QA-OutOfMemory	DatabaseMemoryUsagePercentage > 90 for 2 datapoints within 10 minutes	IPPC-and-EPIC
AWS Service: EC2		
ippc-smtp-qa-StatusCheckFailed	StatusCheckFailed >= 0.99 for 1 datapoints within 5 minutes	When in alarm, reboot the instance with id "i-0624986412f06f55f"

Budget Alerts

AWS Service: AWS Budgets		
CloudWatch	When your actual cost is greater than **110% ($18.92)** of your **budgeted amount ($17.20)**, the alert threshold will be exceeded.	BlillingAlarms
DynamoDB	When your actual cost is greater than **110% ($11.66)** of your **budgeted amount ($10.60)**, the alert threshold will be exceeded.	BlillingAlarms
EC2	When your actual cost is greater than **110% ($365.20)** of your **budgeted amount ($332.00)**, the alert threshold will be exceeded.	BlillingAlarms
EC2-Other	When your actual cost is greater than **110% ($231.00)** of your **budgeted amount ($210.00)**, the alert threshold will be exceeded.	BlillingAlarms
ElastiCache	When your actual cost is greater than **110% ($19.61)** of your **budgeted amount ($17.83)**, the alert threshold will be exceeded.	BlillingAlarms
ELB	When your actual cost is greater than **110% ($12.51)** of your **budgeted amount ($11.37)**, the alert threshold will be exceeded.	BlillingAlarms
RDS	When your actual cost is greater than **110% ($144.10)** of your **budgeted amount ($131.00)**, the alert threshold will be exceeded.	BlillingAlarms
S3	When your actual cost is greater than **$200.00 (111.11%)**, the alert threshold will be exceeded.	BlillingAlarms

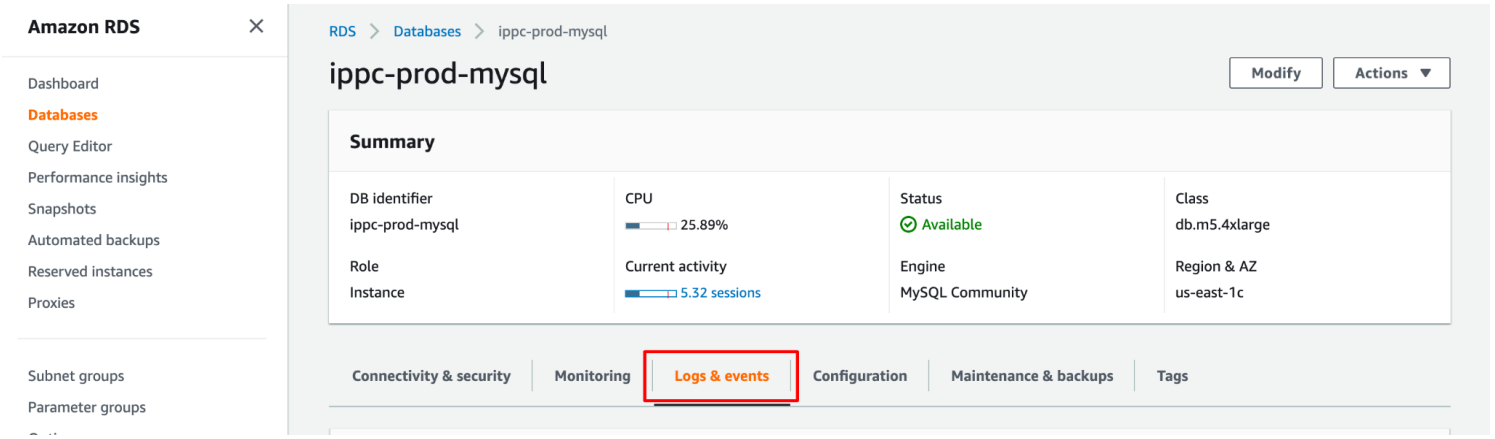
List of planned monitoring activities daily/weekly/monthly/long-term

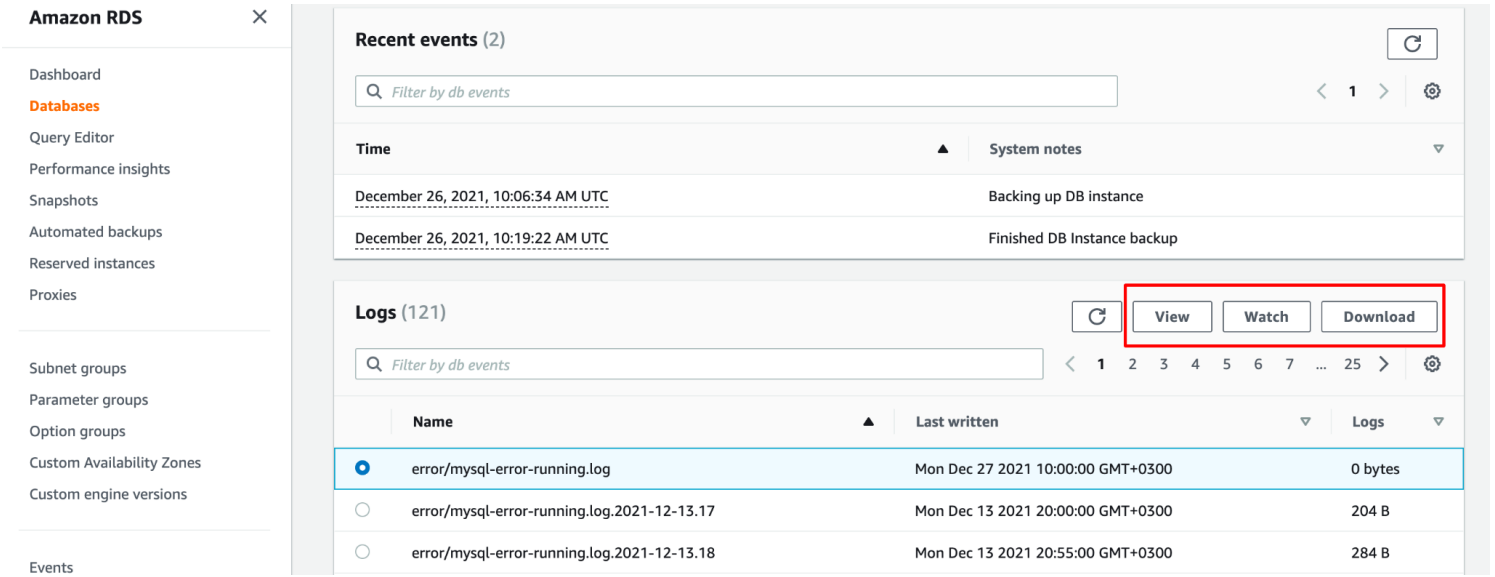
1) Daily monitoring email for the Cloudwatch/Budget notifications and take appropriate actions if they are required(i.e. extend disk space; check logs on the server; restart services)
2) Verifying the result of the **legacy-prod-flag** backup instance and restart process every Saturday morning.

Information related to logs, apart from cloud-watch

RDS

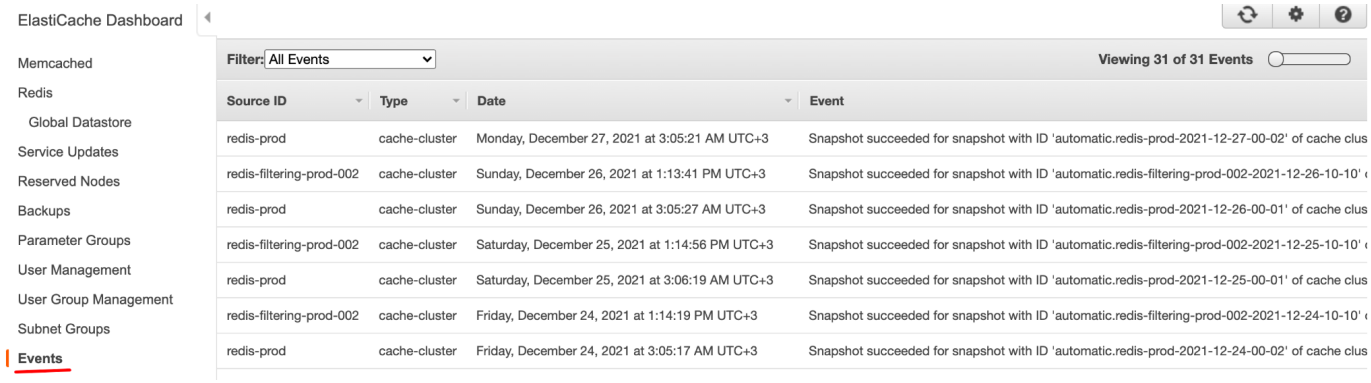
RDS logs and events can be extracted or reviewed in appropriated section:





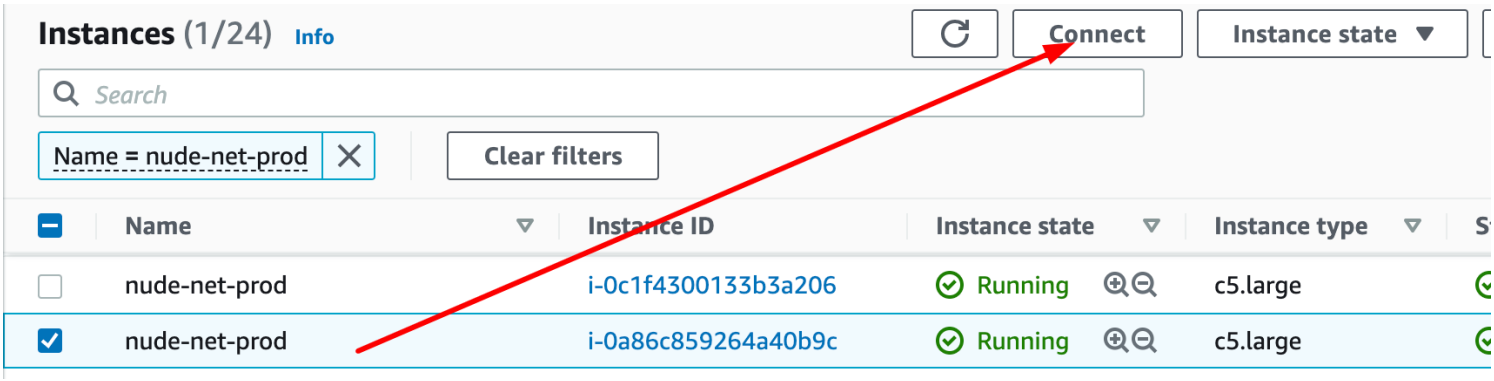
ElastiCache

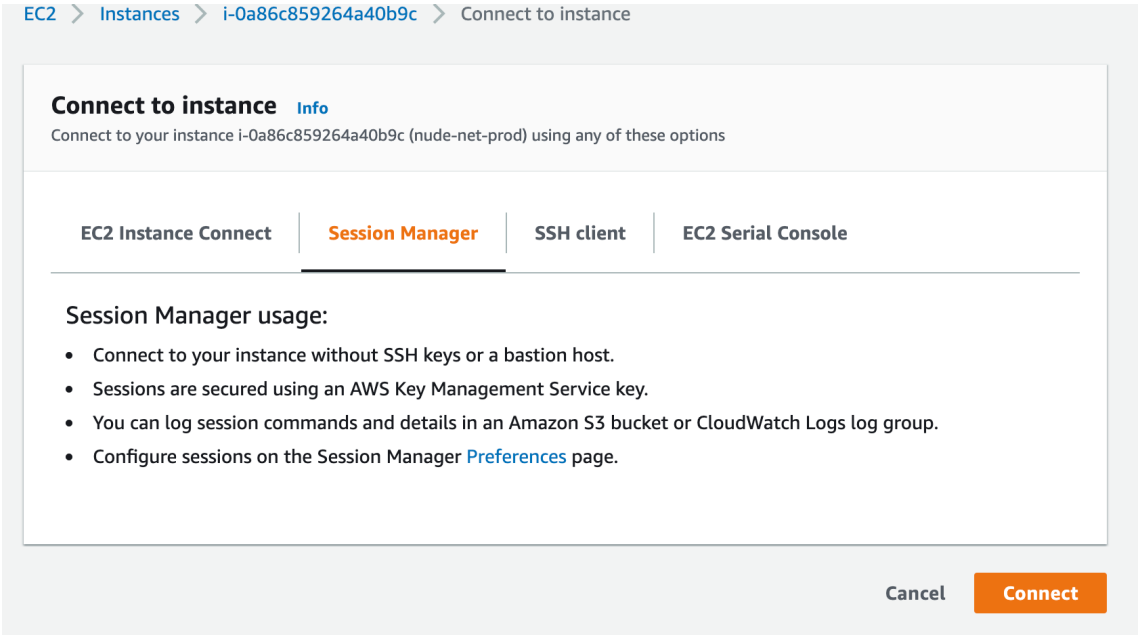
ElastiCache event records can be found in ‘Events’ sections:



Beanstalk instance direct connect

In order to obtain the logs or watch the application logs in real-time we should login to instance using SSH or SSM session:





Get container ID: **sudo docker ps -a**

```
[root@ip-172-31-74-129 /]# sudo docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
50e34ed55874        5b2a2cb53e7f        "python nude_detect.…"   6 days ago          Up 2 days
[root@ip-172-31-74-129 /]#
```

Watch the logs using: **sudo docker logs -f ID**

```
[root@ip-172-31-74-129 /]# sudo docker logs -f 50e34ed55874
TAG[{"label":"FACE_F","score":0.4767853319644928},{"label":"FACE_F","score":0.46843773126602173},{"label":"FACE_M","score":0.3047319352626
8005},{"label":"FACE_M","score":0.28838467597961426},{"label":"FACE_M","score":0.20763853192329407},{"label":"FACE_F","score":0.1978210508
8233948},{"label":"COVERED_FEET","score":0.1700827181339264},{"label":"FACE_F","score":0.14990460872650146},{"label":"EXPOSED_FEET","score
":0.1446147859096527},{"label":"FACE_F","score":0.13175690174102783},{"label":"FACE_M","score":0.13097867369651794},{"label":"EXPOSED_BREA
ST_F","score":0.1254798173904419}]
FACE: {"label":"FACE_F","score":0.4767853319644928}
BODY: {"label":"NONE","score":0.0}
TAG[{"label":"COVERED_FEET","score":0.12117382884025574},{"label":"COVERED_FEET","score":0.10462453961372375}]
FACE: {"label":"NONE","score":0.0}
BODY: {"label":"NONE","score":0.0}
TAG[{"label":"EXPOSED_BREAST_F","score":0.15904313325881958},{"label":"EXPOSED_BELLY","score":0.15676414966583252}]
FACE: {"label":"NONE","score":0.0}
BODY: {"label":"NONE","score":0.0}
TAG[{"label":"COVERED_BREAST_F","score":0.11290132999420166}]
```

In order to read the all records for specific containers find the log location with command:
sudo docker inspect --format='{{.LogPath}}' ID

```
[root@ip-172-31-74-129 /]# sudo docker inspect --format='{{.LogPath}}' 50e34ed55874
/var/lib/docker/containers/50e34ed5587483ec2e4b70511342f9837ca554acb183c5ed0c13c827d73654cd/50e34ed5587483ec2e4b70511342f9837ca554acb183c5ed0c13c827d73654cd-
json.log
```

Additionally If application doesn't write the logs in stdout they can be found directly in the container:
Jump inside with the command **sudo docker exec -it ID bash**:

```
[root@ip-172-31-74-129 /]# sudo docker exec -it 50e34ed55874 bash
root@50e34ed55874:/# cd /var/log/
root@50e34ed55874:/var/log# ll
bash: ll: command not found
root@50e34ed55874:/var/log# ls -la
total 220
drwxr-xr-x  3 root root   4096 Dec  2 03:41 .
drwxr-xr-x 11 root root   4096 Dec  1 00:00 ..
-rw-r--r--  1 root root  11771 Dec  2 03:41 alternatives.log
drwxr-xr-x  2 root root   4096 Dec  2 23:37 apt
-rw-rw----  1 root utmp      0 Dec  1 00:00 btmp
-rw-r--r--  1 root root 158130 Dec  2 23:37 dpkg.log
-rw-r--r--  1 root root   3232 Dec  1 00:00 faillog
-rw-r--r--  1 root root    484 Dec  2 03:41 fontconfig.log
-rw-rw-r--  1 root utmp  29492 Dec  1 00:00 lastlog
-rw-rw-r--  1 root utmp      0 Dec  1 00:00 wtmp
root@50e34ed55874:/var/log#
```

VPN

PPTP VPN runs on an EC2 Linux instance (`legacy-prod-vpn`, in Ohio region).

IP: 52.14.176.136

EPIC credentials: epic / EpicPass86
IPPC credentials: ippc / VPeQct52ciWd8

Coding

Code-level comments

When coding against the NextGen database keep in mind that the database is very large.

There are huge tables in the database, you should not execute SQL joins on those tables.

Recommended way is to make separate requests: first get the list of IDs and then get data based on those IDs.

SQL to get large tables:

```
select
      `table`,
    max(is_event_id) has_event_id,
      format(max(`count`), 0) `count`,
    format(max(size) / 1024 / 1024 / 1024, 2) `size GB`,
    format(max(data_free) / 1024 / 1024, 2) `free MB`
from (
     select
          c.`table_name` `table`,
        c.`column_name` `column`,
           case when c.`column_name` = 'EventID' then 1 else 0 end as
`is_event_id`,
        t.table_rows as `count`,
        t.data_length + index_length as size,
        t.data_free
      from information_schema.`columns` c
    inner join information_schema.`tables` t
          on t.`table_name` = c.`table_name`
    where table_type = 'BASE TABLE'
          and t.table_schema = 'IPPC') t

group by `table`
order by size desc, `table` asc;
```

Outdated code

Some tools are used just once or used rarely.

3rd party APIs and libs.

IronOCR (https://ironsoftware.com/csharp/ocr/) used for OCR needs in Image Analysis process
Reincubate API (https://reincubate.com/ricloud-api/) used to get iCloud backups for iOS devices
MS API (https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision/overview-image-analysis) used to detect adult content on images
YouTube API (https://developers.google.com/youtube/v3) used to get video titles by url id
AWS SDK for S3,DynamoDB,CloudWatch Logging, Secrets.
Pipefy API - to move device cards
Chargify API - to pay for access to NextGen system
Legacy System API - used to create new case/device
Mime.Kit - to parse emails

NextGen Database maintenance

MySQL 5.6 to 8 migration

AWS announced EOL for MySQL 5.6 RDS instances and migration of those to 5.7. A decision was made by IPPC to migrate to MySQL 8.

New MySQL 8 instance with an empty schema was created and data was migrated to that instance. Only 6 months of data were migrated, the old database is now terminated, but data remains available as a snapshot in RDS and can be restored as an instance.

Related JIRA ticket: https://ncptc.atlassian.net/browse/NEX-689

Issues

Missing data

Data was migrated with AWS Data Migration Service. A portion of data was missing in the database which made the system unusable for the Officers. Random data checks which were a part of data migration did not detect the issue.

Data was re-synced from the old database, a tool was created to verify key tables, currently the data is restored and verified in the new database.

Slowness

Based on the user feedback the new database performs slower than the old one. A suite of load tests and performance tests were not run.

At this time the best option to improve performance may be to recreate indexes. 2 indexes have been recreated already. Note - the largest index took 2 hours to get deleted and it took 8 hours to create it again.

Old database

The old database has been terminated to avoid unnecessary costs but it remains available as a final snapshot in RDS (arn:aws:rds:us-east-1:967956968559:snapshot:ippc-prod-final-snapshot), it can be restored within 15 minutes as a new instance if requested by IPPC.

Data clean-up

Contractually IPPC has to maintain 6 month of data online. The more data in the database the worse the performance will be.
A decision was made to implement a process that would remove old data from the database and archive it for future use.

Currently the process is partially implemented.

Suggested process

1) Create a snapshot of the database in RDS (that will not be removed by retention policy)
2) Delete data older than 6 months (a tool was implemented, the code is in `data-cleanup` repo in AWS CodeCommit in Virginia)
3) Optimize database

Clean-up tool

Clean-up tool uses EventID condition (EventID is lower than) for the majority of tables, exact value should be defined based on tblEvent content (just find EventID that corresponds to a certain date). Deleting data from all tables by date is practically not realistic because of potentially poor performance.

There is one table that doesn't have EventID, this table is cleaned up based on date value - tblMachineHealth.

ToDo

To make the process work the team will need to test how long data deletion takes and check if there is a need to rebuild indexes after data deletion to avoid impact on performance.

As a second step, this process can be automated.

Backups

Database backups schedule/rules

Name	Region	Engine	Backup window	Backup retention period	Backup type
ippc-prod-mysql	us-east-1	RDS: MySQL 8.0.23	09:56-10:26 UTC	8 days	System automated snapshot
ippc-qa-mysql	us-east-1	RDS: MySQL 8.0.23	10:20-10:50 UTC	7 days	System automated snapshot
mobistealth	us-east-1	RDS: MySQL 5.7.34	08:00-08:30 UTC	1 day	System automated snapshot
legacy-prod-flag	us-east-2	EC2: MSSQL	At 1:00 Every Saturday	31 day. Rotated by lambda rotate-weekly-flag-snapshots	Created by script on legacy-prod-wood instance C:\scripts\consistent-snapshot.ps1

Additional Resources:

How to test IPPC NextGen

Configuration and environments

Helpful links can find in this Wiki page
https://quicklinks.monitorsolutions.net/ (ippcadmin:ippcadmin2019)

Environments:

Name	NextGen	Legacy
Testing	qa-dashboard.monitorsolutions.net	https://qa.inetppc.com/IPPC/frameset.htm
Staging	stage-dashboard.monitorsolutions.net	https://qa.inetppc.com/IPPC/frameset.htm
Production	dashboard.monitorsolutions.net	https://monitor1.inetppc.com/IPPC/frameset.htm

Credentials to sign in:

Testing	Epic:EpicTesting
Staging	Epic:EpicTesting
Production	Epic2:Epic2

Test artifacts

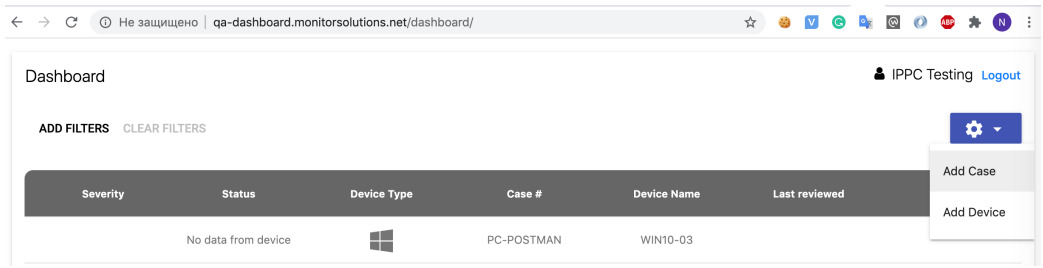
Regression Tests - NextGen

The regression should be run before each release on the stage environment to be sure everything is ok. The checklist is divided into different types of devices and a common part. Before running tests, you must always create a new case and install all devices from scratch to be sure all parts (creating case, creating device, payments, pipefy cards, chargify) are covered. We are using a checklist also for the smoke testing, but everytime score should be defined by QA depends on functionality which is deploying to production.

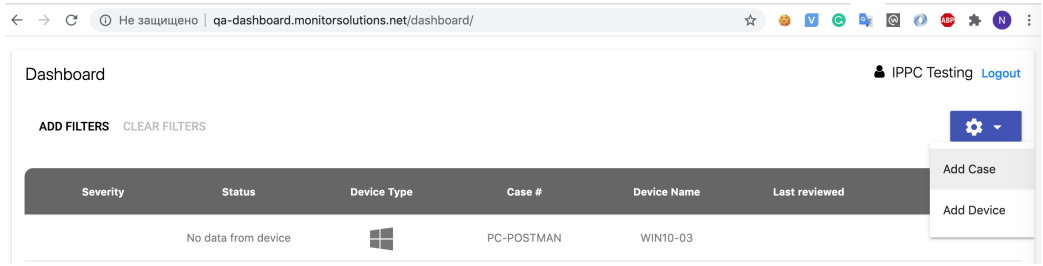
How to install PC toolkit

1. Setup the case
    a. Open https://dashboard.monitorsolutions.net/dashboard/ , https://qa-dashboard.monitorsolutions.net/dashboard/, https://stage-dashboard.monitorsolutions.net/dashboard/

    b. Add case

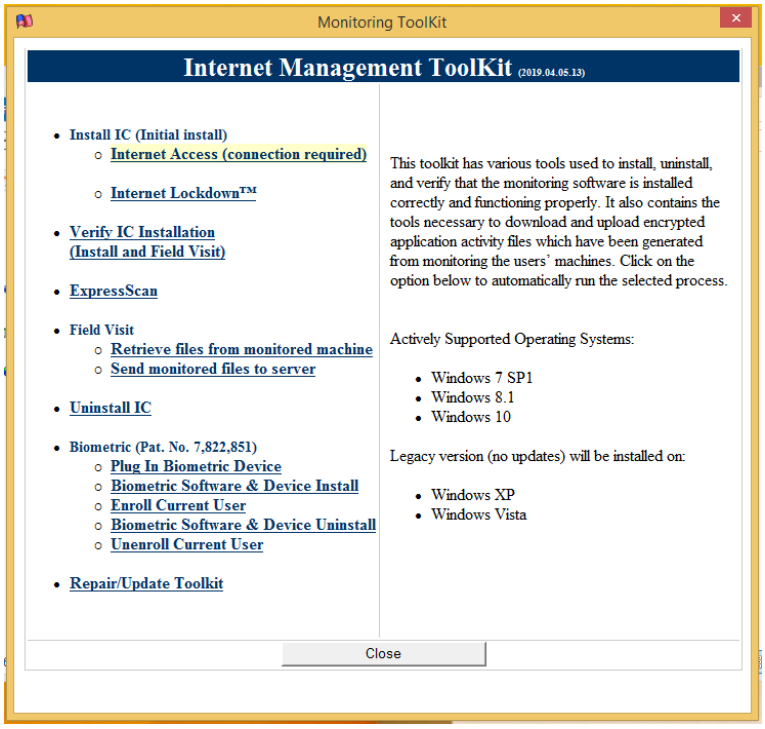


2. Setup the device
    a. Add a device

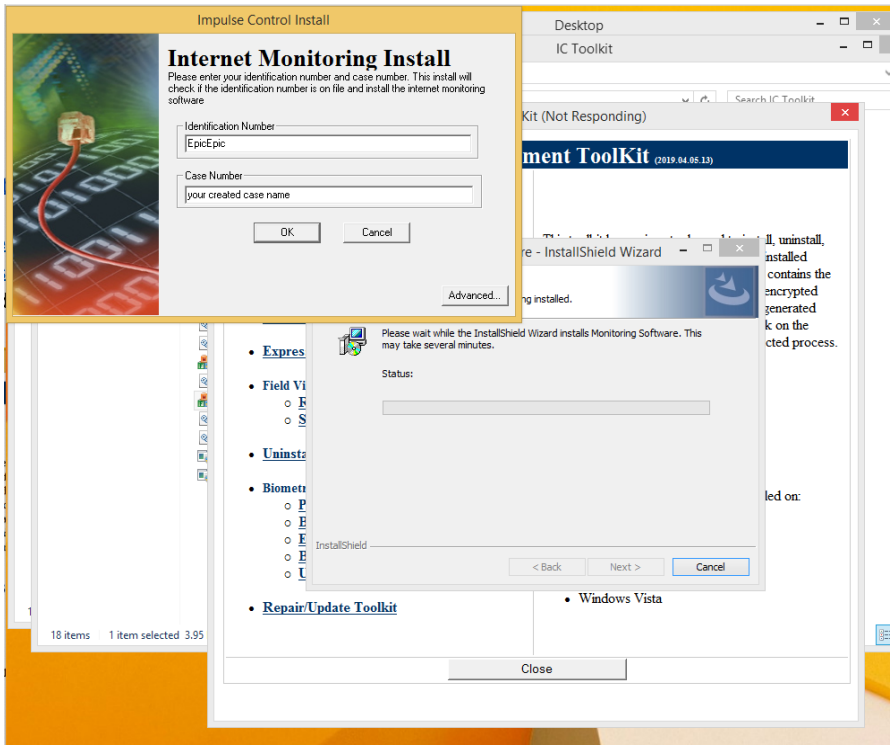


    b. Get the Email with Payment setup and proceed with instructions
    For the QA you can use the credit Card 4111 1111 1111 1111
    c. Get the Email with appointments and it MUST be opened and clicked a button.

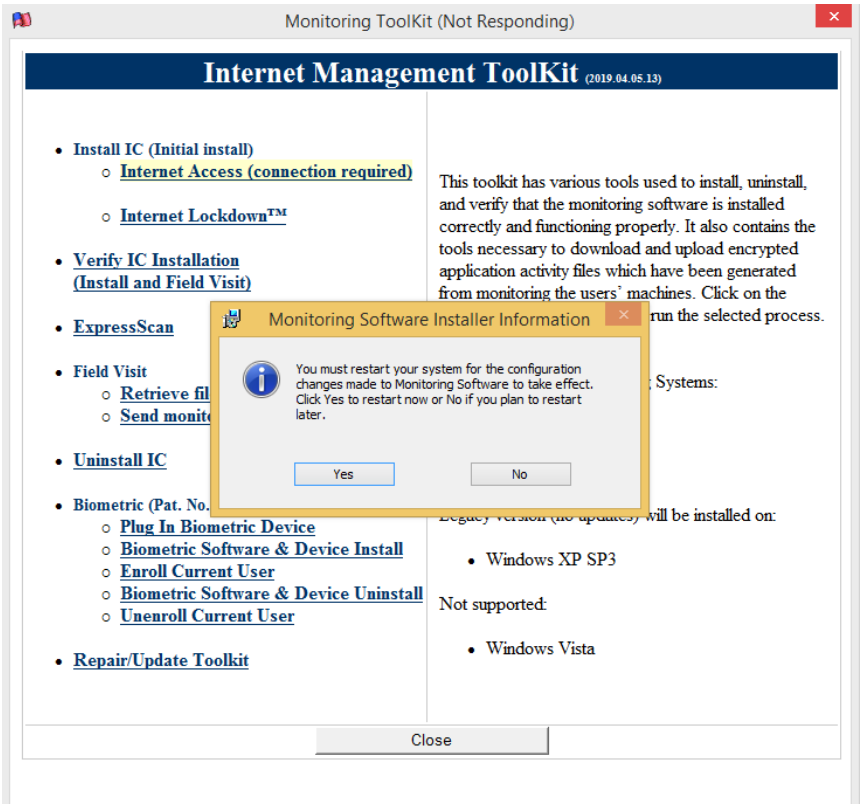
3. Install a toolkit
    a. Turn off UAC on your device
    b. Download (ask Jeff Metzner <jmetzner@ippctech.net> for the latest version) the toolkit https://ippc-bucket-downloads-private.s3.us-east-1.amazonaws.com/IC%20Toolkit%202020-07-01%20%282020%20pre-release%29.zip
    c. Run MT10.exe under Administrator rights and click Internet Access (connection required)



d. Enter EpicEpic and the case number which you created, in case you want to use the QA click Advanced and enter the IP address 34.195.213.182 in both fields.



e. Restart your PC



Test clients for Android and Windows are located here
https://drive.google.com/drive/folders/1q6phhW4861POHjqw81HYGePGG2Uv6CTJ?usp=sharing

Reports and Report Schedule

Here is a list of the reports that are automatically generated and sent by IPPC backend processes and their frequencies

Report Name	Description	Process location	Frequency
Officer's Dashboard	Each Officer receives an email each Monday with how their NextGen Dashboard looks like. This helps the Officers get a quick look at Cases with High and Medium severity events reported in the last week. Also, it includes the status for Heart Icon and Payment		Weekly on Mondays
Smart Images	Each Officer receives an email each Monday with a summary of the images that were flagged during the week with the Image Analysis process		Weekly on Mondays
Payment Report	Officers and Supervisors receive an email once a month with a summary of all the Cases assigned to them and the status of the payments as they are reported in Chargify.		Monthly on the 1st day of the month
Case Termination	This process runs daily to check which Cases have defined a termination date that is set 90, 30 or 3 days from today. The Officers are notified on those days that the Case is close to be automatically terminated		Daily (the report is only sent to the Officers that have Cases ending in 90, 30, or 3 days.
Risk Words Pending Approval	This report is intended for IPPC Admin personal to receive a list of Risks Words that are pending to be approved.		Daily As of December 23, this report has not been approved to deploy to Production Reference:

Ongoing Projects and Issues (December 2021)

Known critical bugs and issues

This is a list of issues as of December 23, 2021 that are critical

Issues and Related Tickets	Description	Status
Database optimization after migration to MySQL 8 Improve DB performance via Indexes	As of December 23, 2021 there are questions about the new Production database recently migrated from MySQL 5 to MySQL 8. This needs a closer look to see if there are issues and the root cause of these issues.	In Progress
Reprocess lost data after AWS outage Reprocess lost Android and iOS data	As of December 23, 2021 there is an ongoing process to be able to recover from an AWS outage that caused losing the events that were in the Redis cache queues waiting to be processed.	In Progress
YouTube quota issue Modify URL filtering process for YouTube YouTube - Workaround link conversion when quote is exceeded YouTube - Cache YouTube information	IPPC is having issues querying the YouTube API to provide the details that are needed to convert Web events into YouTube events in a correct way. The issues are related to IPPC exceeding the daily quota assigned by YouTube to query for information. The tickets listed here are solutions proposed to reduce the number of calls to the YouTube API and resolve the quota issue	Development done, tickets waiting to be tested and released in Production
YouTube link conversion issues YouTube - Modify Android YouTube link conversion YouTube - Remove suffix information from Android YouTube links Windows - YouTube video not getting the correct title	There have been some issues reported related to the conversion of Web events into YouTube events. The issues in this case are related to YouTube not providing accurate results or not providing results at all.	Development done, tickets waiting to be tested and released in Production
Slow performance in NextGen Reports generation section **Timeout loading Swimlanes view in NextGen**	Given the amount of data that exists in the database and the fact that some tables may not be optimized for querying data, causing performance issues and	Development in progress. Data Warehouse table created in QA undergoing

Data-warehouse tables and sub-tasks associated to this task NextGen - Timeout loading swimlanes	timeouts, we have been working on the creation of a new structure optimized for querying for data. After the new table is loaded and optimized, the NextGen Swimlanes view and NexGen Reports tab will need to be changed to utilize the new structure to obtain the data.	testing and bug fixing. Pending definition of indexes to be applied to the new table. Working on changes in NextGen report to utilize the new structure. Pending working on changes in NextGen Swimlanes view to utilize the new structure.

IPPC Release Procedure and Guidelines

There are three types of releases that are typically performed on the IPPC project. Type of a release is determined by what components are included in the release, and by the urgency of the release.

In general, among the many components that the comprise the IPPC NextGen system, the “UI Dashboard” (aka Officer Dashboard) component is the one that is facing the users, so, if included in a release, it usually requires full pre-release and post-release regression testing.

The types of releases are:

1. A “scheduled” or “full” or “normal” release — this is a release that is usually performed every month, (preferably at the beginning or in the middle of a month, due to some of the team members’ workload schedule) it includes changes to the Officer Dashboard component, and is normally accompanied by a full pre-release (performed on the Staging environment) and a post-release (performed on the Production environment) regression testing.
2. A “non-UI” release. This is a type of release that doesn’t include changes to the Officer Dashboard component. This release can be performed at any time, since it usually doesn’t require full regression testing. Depending on the scope of the release, it might include a smoke test of this or that area of the system.
3. A hotfix release. This release is usually performed either after a critical production bug is found or immediately after a scheduled release, during the regression testing for each, critical issues were found. During this type of release the pre-release testing might be omitted, per internal agreement with all the team members.

The scheduled release is performed according to a checklist.

Here’s an example of a release checklist template:

	IPPC	Release 16				
		Checklist				
		Step	**Status**	**Do on**	**Result**	**Comment**
	1	Confirm release scope (tickets)	Pending			
	2	Confirm all tickets scheduled for the release can be merged easily	Pending			
	3	Confirm that mail hog is operational	Pending			
	4	Confirm all Jira components set for release tickets	Pending			
	5	Estimate time to perform the release	Pending			
	6	Decide on regression environment (QA/Staging, Staging)	Pending			
	7	Plan appropriate buffer for QA and regression	Pending			
	8	Verify git branch name field is filled out for all tickets	Pending			
	9	Verify Deployment Dependencies field is filled out for all tickets	Pending			
	10	DevOps to verify configuration changes	Pending			
	11	QA all release tickets (move to the "Ready for Release" state)	Pending			
	12	Resolve all issues found in QA, or exclude a ticket with an issue	Pending			
	13	Prepare a release branch & merge all tickets	Pending			
	14	Deploy release branch for regression testing	Pending			
	15	Monitor queues in QA during regression	Pending			
	16	Perform regression testing	Pending			
	17	Verify mail hog every day	Pending			
	18	Resolve issues found during regression testing	Pending			
	19	Do we need to perform full regression?	Pending			
	20	Receive formal approval from IPPC	Pending			
	21	DevOps to implement config changes	Pending			
	22	Run database migration scripts	Pending			
	23	Deploy release branch to production	Pending			
	24	Perform post-release regression	Pending			
	25	Update test documentation	Pending			
	26	Move all tickets not included in the release to the "Needs Testing"	Pending			

Scheduled Release Checklist Steps

1. Confirm release scope (tickets)
   Product owner(s) confirm the targeted scope of the release (list of included tickets)
2. Confirm all tickets scheduled for the release can be merged easily
   For all tickets that are scheduled for the release, one must confirm that their code changes are relatively recent, and can be merged in the master branch. This is because sometimes the changes implemented in the code on a feature branch can sit for quite some time, and the feature branch diverges too much from the master branch. When this happens, it takes a lot of time to merge the feature branch and resolve all conflicts. Often it is easier to re-write the code on the feature branch from scratch. Hence this step to verify that the feature branch of the ticket that is targeted for the release scope can be merged into the master branch relatively easily.
3. Confirm that mail hog is operational
   Someone on the dev/devops team confirms that the MailHog (the component configured to test emails) is working properly
4. Confirm all Jira components set for release tickets
   For each Jira ticket included in the release, make sure that the “Component” field is set. This is required for the release deployment, to know exactly which components need to be redeployed
5. Estimate time to perform the release
   How long will it take to release and deploy all components included in the release. Sometimes it takes an hour, sometimes several hours.
6. Decide on regression environment (QA/Staging, Staging)
   Decide on whether the Staging environment will be used to deploy either the entire system, or just the Officer Dashboard component. Default is the latter.
7. Plan appropriate buffer for QA and regression
   If tickets scheduled for the release are still in the “Testing” state (not in “Ready for release”), plan time required to move all those to the “Ready for Release” state, taking potential time to resolve last-minute found issues into consideration.
8. Verify git branch field is set for all tickets
   For each ticket, make sure that the “git branch” field is set (or “N/A” is set, if the ticket has no changes to the code). This is required for when the release branch will be prepared.
9. Verify Deployment Dependencies for all tickets
   For each ticket included in the release, make sure that the “Deployment Dependencies” field is filled out, or “No Deployment Dependencies” value is set for that field. This is to ensure that for all tickets, their deployment dependencies are known to the entire team.
10. DevOps to verify configuration changes
    If there are any configuration changes required for a particular ticket (listed in the “deployment dependencies”), DevOps must review the changes and confirm they’re okay.
11. QA all release tickets (move to the "Ready for Release" state)
    If some of the targeted tickets are not in the “Ready for Release” state, give the QA and Dev team time to finish the tickets to bring them all to the “Ready for Release” state.
12. Resolve all issues found in QA
    If any issues are found during QA, they should be resolved before the release is a go-ahead, or if there’s a time constraint, a problematic ticket must be excluded from the release.
13. Prepare a release branch & merge all tickets
    Someone on the dev team prepares the release branch. Then, each individual developer is merging their ticket to the release branch.
14. Deploy release branch for regression testing
    Release branch is then deployed to Staging or QA+Staging (see step 5.) environment, and the QA team starts the pre-release regression testing.
15. During the pre-release regression, all system queues must be monitored to make sure there are no delays in the pre-release regression.
16. Perform regression testing

17. Verify mail hog every day
    Check results produced by MailHog during the pre-release regression, and make sure the number of sent emails conforms to the expected number of emails sent during the performed scenarios.
18. Resolve issues found during regression testing
    If during the pre-release regression testing any issues are found, they must be resolved on the release branch, and verified by the QA team.
19. Do we need to perform full regression?
    Confirm that full post-release regression must be performed.
20. Receive formal approval from IPPC
    IPPC formally approves the release scope, date and the agreed-upon testing (full post-regression, smoke, etc)
21. DevOps to implement config changes
    If there are any configuration changes, they must be performed by the DevOps
22. The person responsible for the release must run the database migration scripts
23. Deploy release branch to production
    Perform the actual code deployment
24. After the person responsible for the release confirms the release is complete, the QA team starts the post-release regression testing.
25. After the QA team is done with the post-release testing, they must update testing documentation if necessary
26. After the release is complete, move all tickets from the “Testing” column (if any are present there) to the “Verify Code” column.