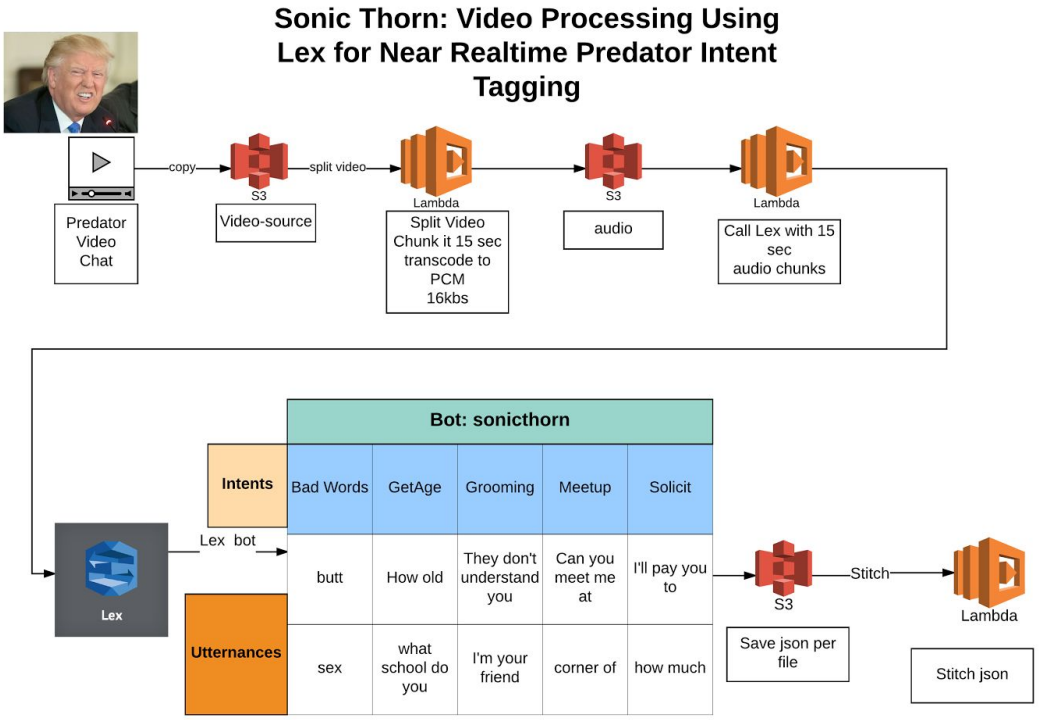
Sonic Thorn

Team members:
Jack jzbruno@gmail.com
Roberto rmonge4surf@gmail.com
Josh josh.andrew.phillips@gmail.com
Jessica jessica.j.major@gmail.com
Chris zonk1024@gmail.com

https://sonicthorn.signin.aws.amazon.com/console
@awsreinvent2017

Git Rep

- git@github.com:redstonemercury/sonic-thorn.git
- https://github.com/wearethorn/aws-plugin



Pipeline

- S3 In -> split_audio -> save to s3 audio bucket -> chunk audio into 15 sec units->save to S3_audio -> Lex to Transcode audio to txt and find intent -> save json->stitch-output

=========

Transcode -> split audio

- This could be good for pulling audio https://www.ffmpeg.org/ or look at amazon transcoding service and see if you can pull out
- Ffmeg can split files out into 15 second files
    - https://unix.stackexchange.com/questions/280767/how-do-i-split-an-audio-file-into-multiple
    - ffmpeg -i somefile.mp3 -f segment -segment_time 15 -c copy out%03d.mp3
- Console services elemental
    - https://www.elemental.com/newsroom/press-releases/aws-announces-family-five-aws-media-services-complete-video-workflows
    - Elemental and Transcode did not have options to split out audio to 16khz Linear PCM which Lex required. Had to had setup with ffmpeg.

Audio -> Speech

- Channing triggers
    - S3 bucket upload
    - Triggers Lambda to split out audio but there are size limiations
    - Google has an async functiont that would have been very helpful https://cloud.google.com/speech/docs/async-recognize
        - We wouldn't have had to break down the audio into smaller chunks.

- Amazon Lex
    - Example : http://docs.aws.amazon.com/lex/latest/dg/gs-create-test-speech.html
    - http://docs.aws.amazon.com/lex/latest/dg/API_runtime_PostContent.html The `PostContent` operation supports audio input at 8kHz and 16kHz. You can use 8kHz audio to achieve higher speech recognition accuracy in telephone audio applications.
    - First make an audio file
        - (This works, need the pcm format for lex) aws polly synthesize-speech --region us-east-1 --output-format pcm --text "This is very bad text. I am a bad person. Please find me and arrest me. I am very sick. " --voice-id "Joey" audio_thorn_pcm.raw
            - You can play it with
                - ./ffplay -f s16le -ar 16k -ac 1 ~/audio_thorn_pcm.pcm
            -
        - (This works, gives back “inputTranscript” in output that contains the transcript) aws lex-runtime post-content --region us-east-1 --bot-name sonicthorn --bot-alias ”\$LATEST” --user-id UserOne --content-type "audio/l16; rate=16000; channels=1" --accept "text/plain; charset=utf-8" --input-stream audio_thorn_pcm.mpg IntentOutputSpeech.mpg
    - Justin from Thorn said that they also use GCP so we could stub out the GCP option.
    - Slice it up, get it back.
    - http://docs.aws.amazon.com/lex/latest/dg/gl-limits.html
    -
- https://pypi.python.org/pypi/SpeechRecognition/ (If we have time)

Demo Ideas

- Record a Video using key trigger words
- Push to s3 source bucket
    - This kicks off workflow
- Show diagram discuss arch
- Come back to results page.

- Summary audio.

Future Ideas:

1) Make an SRT file with subtitles and mark the bad ones with some kind of timecode. .

Future idea:
.srt file format
https://en.wikipedia.org/wiki/SubRip

1 00:00:22,000 --> 00:00:27,000 I'll teach thee Bugology, Ignatzes
2 00:00:40,000 --> 00:00:43,000 Something tells me

once the .srd is made (by lex returns) we could add them back into the main large file
https://trac.ffmpeg.org/wiki/HowToBurnSubtitlesIntoVideo

2) Machine Learning

of full output and do more sophisticated NLP with more context.

- See https://github.com/ceteri/pytextrank/
- Hook into existing UI or at least identify how to hook up to app.