Déroulement de la conférence "BlendWebMix"

Préparation :					
	S3	Supprimer le bucket (créé pour réserver le nom)			
	*	Ouvrir tous les fichiers de config dans un éditeur de texte			
	AWS	Ouvrir le compte + Vérifier la région (Ireland)			
Conférence :					
Démo	**Service**	**Opération**	**Config**	**Fichier**	**A sauver**
1	**S3**	Créer bucket	Nom : serverless-bwm-labyrinth		Nom du bucket
		Upload fichiers + make public	Fichiers fournis (sauf répertoire "aws-config")	répertoire code	
		Activer l'hébergement statique (onglet properties)	Index : index.html - Pas d'erreur		URL d'accès
		Activer le CORS (onglet permissions)	Laisser la config par défaut	(backup dans awsConfig/s3/CORS.xml)	
	Site	Raccourcir l'URL et la donner			
		Tester le site - pages statiques OK			
Démo	**Service**	**Opération**	**Config**	**Fichier**	**A sauver**
2	**Cognito**	Créer user pool (étape 1)	Nom : ServerlessLabyrinth puis review defaults		
		Changer la password policy	(simplifier pour la présentation)		
		Valider et créer la user pool			Pool Id
		Créer l'app clients	Nom : ServerlessLabyrinthApp / Uncheck "Generate client secret"		App client id
		Changer le fichier config.js => y mettre le pool id et le app client id			
	S3	Upload du fichier modifié (+ make public)			
	Site	Redonner l'URL			
		Tester : créer un utilisateur (register - verify - signin)			
Démo	**Service**	**Opération**	**Config**	**Fichier**	**A sauver**
3	**DynamoDB**	Créer table	Nom : Labyrinth, Clé : key (string)		
		Aller dans les capacités pour montrer l'autoscaling (si besoin)	Mettre 25/25		
		Dans items, créer un nouvel item		awsConfig/dynamoDB/Labyrinth_item.json	
	IAM	Créer un rôle	AWS Service Role -> AWS Lambda puis policy "AWSLambdaBasicExecutionRole", nom : serverless-lambda, description : Lambda role for the serverless labyrinth		
		Ajouter une "inline policy" au role	Via policy generator : putItem + getItem sur DynamoDB	(custom si besoin : awsConfig/iam/rolePolicy.json)	
	Lambda	Créer une fonction lambda (blank) sans trigger	Nom : serverless-GetGridState, Runtime : Python 3.6, Role : serverless-lambda, Code : fichier	awsConfig/lambdas/GetGridState.py	
		Tester	Contenu du test : {} (vide)		
		Créer une fonction lambda (blank) sans trigger	Nom : serverless-MovePawn, Runtime : Python 3.6, Role : serverless-lambda, Code : fichier	awsConfig/lambdas/MovePawn.py	
		Tester	Contenu du test : fichier	awsConfig/lambdas/test_MovePawn.json	
		Créer une fonction lambda (blank) sans trigger	Nom : serverless-TurnBlock, Runtime : Python 3.6, Role : serverless-lambda, Code : fichier	awsConfig/lambdas/TurnBlock.py	
		Tester	Contenu du test : fichier	awsConfig/lambdas/test_TurnBlock.json	
Démo	**Service**	**Opération**	**Config**	**Fichier**	**A sauver**
4	**API Gateway**	Créer une nouvelle API	Nom : ServerlessLabyrinthAPI		
		Créer un authorizer => create => cognito user pool	Région : eu-west-1, user pool : ServerlessLabyrinth, token : Authorization, validation : vide		
		Créer une nouvelle méthode GET dans /	Type : GET, région : eu-west-1, function : serverless-GetGridState		
		Ajouter l'authorizer à GET (clic sur Method Request)	Authorization : choisir le user pool ServerlessLabyrinth		
		Créer une nouvelle méthode POST dans /	Type : POST, région : eu-west-1, function : serverless-TurnBlock		
		Ajouter l'authorizer à POST (clic sur Method Request)	Authorization : choisir le user pool ServerlessLabyrinth		
		Créer une ressource	Nom : pawn		
		Dans /pawn, créer une nouvelle méthode POST dans /pawn	Type : POST, région : eu-west-1, function : serverless-MovePawn		
		Ajouter l'authorizer à POST (clic sur Method Request)	Authorization : choisir le user pool ServerlessLabyrinth		
		Activer CORS : action - enable CORS	**Sur / ET sur /pawn**		
		Déployer (action - deploy api)	Stage: new (prod)		Invoke URL
	S3	Modifier Config.js => y rajouter l'url de l'api			
		Upload du fichier sur S3 (+ make public)			
	Site	Test !! :)			