GameLift Launch Check List (Korean)
20.07.21

개발 단계

	확인 항목	원문
1	게임 클라이언트(실제 게임이 실행되는 머신)가 직접 AWS SDK 호출을 통해 Gamelift 서비스에 접근하지 않도록 설계	Check that the clients (game executable running on any player machine) are not making AWS SDK calls directly to GameLift. Requests should be authenticated through other AWS infrastructure, ask us for help if this sounds like gibberish.
2	GameLift Fleet 서버 내에서 다른 AWS 서비스 접근을 위해, 1. IAM Access key 대신 IAM Role 사용 권장 2. IAM Role 내에 필요한 최소한의 권한만 부여	Make sure that you are carefully managing any keys or secrets used by the servers, principle of least privilege (POLP) is observed, and a rotation schedule is in place. Compromised server processes are unlikely, but such processes definitely shouldn't have access to the keys. The best practice is to use IAM Roles to give GameLift server instances access to other AWS resources, and this should be mandatory.
3	1. 게임 서버에서 별도로 EBS 에 로그를 쌓는 경우 EBS 볼륨이 가득 차는 것을 방지하기 위해 주기적으로 지워질 수 있도록 관리 필요. 2. 게임 서버가 중단될 경우 로그가 유실되지 않도록 별도의 로그 관리 방안 수립 필요 (Cloudwatch Log service, ELK stack, Splunk, LogStash, Flume 등)	If your server writes logs to EBS/instance store, check that these are eventually deleted (*e.g.* an hour or so later), even if they are collected and uploaded by GameLift. Use a different directory or log file name for each server process and each game session. GameLift does not manage your log file life cycle, and logs can accumulate and exhaust EBS storage on the instance. Also instances can be removed at any time causing you to lose some logs, so expect that too. A better option may be to use the Cloudwatch Log service,

		an ELK stack, or other log aggregation service like Splunk, LogStash or Flume. Standardize the log data so that it can be read and understood in context, in real time, and it is easy to set alarms for different failure conditions.
4	대상 리전에 Queue, Fleet 배포 계획 수립 및 자동화	Plan the roll-out of fleets to your target regions, with queue and fleet structure. Automate the deployment process.
5	로드 테스트를 위한 테스트 클라이언트 준비 및 자동화	Build a load test client. Ideally, automate some level of load testing too.
6	매치 메이커 및 Queue 를 하나 이상의 리전에 배포하여 가용성 확보. (만약 특정 리전 / AZ 에서 이슈 발생시 다른쪽으로 fail over 되도록 구성)Queue 의 경우 동일한 Destination fleet 을 여러 Queue 에 할당할 수 있음	Make sure that there is a matchmaker and a queue in more than one region. These are not redundant by themselves. The fleets can be destinations on both queues. Then if an availability zone or regions is unavailable, and you start getting errors, your back-end should fail over from one matchmaker to the other automatically. That feeling that you experience when that happens and everything continues as if nothing has happened is called operational excellence.
7	알파 / 베타 테스트를 통해 부하를 점진적으로 증가시키면서 테스트 수행 (보통 게임 세션 생성 및 배치 시 부하가 높음) 테스트 수행 시에는 게임 세션 길이, 게임 세션 재활용 비율을 조정 예를 들어 1 분세션길이동안 100K CCU 1v1 의 경우 약 초당 833 새로운 게임	Test your system under gradually increasing loads with alpha and beta testing and soft launches (the whole checklist applies to each of these mini-launches!). Most sensitive to extremely high loads are the Game Session Placement, and Game Session Creation operations. Running longer game sessions, or reusing game sessions for automated rematching, where possible, improves these dynamics considerably. *E.g.* 100K CCU 1v1

	4 분 세션 길이 동안 100K CCU 4v4 의 경우 , 2X 게임 세션 재활용의 경우 초당 13 개의 새로운 게임 생성과 동일	with one minute matches matches/places/creates 833 new games per second (high) 100K CCU 4v4 with four minute matches and 2X reuse of sessions, matches/places/creates 13 new games per second (good). Spread game creation between regions. If you have unavoidably high requirements, please talk to us early on!
8	로그 수집 및 분석 계획 수립	Collect lots of data! Logs and analytics are your best actionable data so think carefully about what you need to collect and how to effectively manage and analyze that data. It will drive operational management, design changes, future technical development, business intelligence, customer satisfaction and product longevity throughout the whole process.
9	고객 CS 계획 수립 및 검증	Define and test the customer support experience for your players. Customers are won and lost here. Empower community managers with the tools to fix customer problems and issues.

런치 준비 (1 달 전)

	확인 항목	원문
1	내부적으로 런치일 결정, 주초(월요일)에 Launch 하는 경우 평일에 대응을 할 수 있는 시간 확보 가능	Select and internally communicate a launch date. Launch early in the week if possible so the first live days are work days.
2	요구사항에 맞추어 Gamelift 를 포함하여 관련 서비스의 리밋 상향	Raise GameLift (and other) service limits to ensure that the live environment can scale up to the

		maximum size, which is at least the sum of the maximum instances for all fleets in the region.
3	System 에 대한 충분한 가시성 확보 여부. 툴 / 커맨드 / 대응 절차 준비 여부 확인	Make sure that your dev-ops have enough visibility into your system, and effective command and control tools and protocols. Imagine that the game controls a nuclear power plant. Document (and laminate checklists for) what you think could fail, and what protocols and procedures will be used to mitigate. These protocols and procedures are sometimes referred to as runbooks. Automate the procedures if possible.
4	커뮤니티 매니저의 런치 준비 상태 확인. (웹 사이트, 소셜 미디어, 포럼 커뮤니케이션)	Make sure that your community managers are ready for the launch and hyping/exciting players through web site, social media, forums etc. in advance of your launch
5	인스턴스당 프로세스 수가 적당한지 점검. 맵 로딩과 같은 작업이 기존에 플레이 중 인 게임에 영향 주지 않는지 확인	Check that the number of servers running on an instance at full tilt (*i.e.* not just quiescent) are within the capabilities of the server instance type. Also, if there are busy activity periods during non-core gameplay, like loading a map, check that one or more games loading (typ. no players attached) doesn't impact running gameplay sessions. Lower process or thread priority from within the game server as necessary during these operations.
6	Fleet 스케일링 정책을 통해 인스턴스가 under provision 되지 않도록 설정. 초반에는 여유 Capacity 를 포함하여 보수적으로 설정하는 것을 권장함. Rule	Don't let scaling policy under-provision instances for the fleet. Tune scaling policy to be more conservative at first, providing a bit more idle capacity that you think you will need. You can

	기반의 스케일링 정책보다 Target-based policy 를 사용하고, 20%의 idle 세션을 확보할 수 있도록 설정하는 것을 권장함	optimize for cost later. Consider the use of target-based scaling policy with 20% idle or more instead of rule-based policy for the best experience.
7	Flexmatch 규칙 내 Latency 기반의 규칙이 포함되었는지 확인 (지리적으로 같은 위치의 플레이어끼리 서로 매칭). 로드 테스트 클라이언트를 이용하여 검증 필요	Check that the Flexmatch rules contain a latency rule (if you are passing latency data) so that players who are logically close to the same region typically play together. Test how this behaves under load with synthetic latency data from your load test client. If you use latency data in one part of the system then use it everywhere it can be used (Flexmatch, Game Session Queues, Backfill requests), or the results are undefined.
8	플레이어 인증과정 및 Infra 가 증가된 로드에 맞추어 스케일 될 수 있는 지 검증	Make sure that your player authentication and infrastructure related to getting players in games (such as your own custom matchmaking) can scale effectively to meet demand. Consider load-testing this mandatory.
9	서버 프로세스가 몇일간 동작을 해도 신규 커넥션을 문제 없이 받아들이는지 검증 필요. 예를 들어 Unity 의 경우 C# TcpListner 를 오랫동안 오픈해 두면 새로운 커넥션을 받지 못하는 경우도 있음.	Check that the server, if left running for several days, is still able to accept connections. For example (unity users), leaving a C# TcpListener open for days will eventually stop being able to receive and respond to connections!
10	모니터링 체계 검증 필요. 게임 리프트 서버 플릿 내 이슈가 있을 경우 알람 발생 여부 및 대응 절차 검증	Set Cloudwatch alarms (if you are using Cloudwatch log service) for any messages that indicate severe or fatal problems on the server fleet. Simulate some failures if practical and test the runbooks.
11	Spot 인스턴스가 해당 리전에서 사용 가능한지 확인, Spot history 그래프 상	Check that spot instances are typically available for the regions and instance types you are using.

	spot 가격이 On demand 의 근방에 다다른다면 spot 인스턴스가 가용하지 않다는 하나의 지표임.	If, in the spot history graph, the spot prices approaches the on-demand price, this is a prime indicator that the spot instances will not be available to FleetIQ. Other historical data may be available in the AWS console.
12	추가적인 확인이 필요한 사항에 대해서는 최대한 빠르게 AWS 에게 문의 요망	Ask us ahead of time for anything you are unsure of. We are here to help and have lots of experience of developers launching.

런치 최종 점검 (1 주일 전)

	확인 항목	**원문**
1	이슈 발생시 대응을 위하여 AWS Support level 이 Business 또는 Enterprise 인지 확인.	Raise your AWS support level to Business or Enterprise level, so that AWS can respond to you in the case of problems or outages outside of office hours. *Your solutions architect has no access to your fleets, and cannot help you if your servers stop working – call AWS Support first.*
2	클라이언트 및 서버 코드 Fix , 최종 QA 검증	Lock down client and server code bases completely and do a full QA suite on the final executables.
3	관련 인원 포함한 War room 구성 (개발자, 운영, 커뮤니티 매니저, AWS 인원)	Establish a war room that is manned (physically and in-person if possible) by the stakeholders (e.g. developers, operations, community/support, production management, business intelligence, executive management) for the period up to and through launch. GameLift management and solutions architects will often participate in your Slack channel during your launch window. Make sure that coffee supplies are adequate, and

		also do some problem solving drills using your protocols and procedures checklists/runbooks.
4	Live 환경 구성. Queue 에 배포 리전당 최소 1 개의 on-demand fleet 배정. (Queue 동작 방식에 대한 이해 필요)	Create your live environment. Check that each queue in use has at least one on-demand fleet in each region that it must serve. Understand the rules for how queues place games (which is different depending on whether there is latency data present, or not), and create your live on-demand and FleetIQ (spot) fleets accordingly, using final executables.
5	라이브 환경을 위해 새로운 Queue 와 매치메이킹 설정파일을 사용하도록 설정. 이를 통해 최신 서버 코드를 사용하고 있는지 다시 한번 점검할 수 있음. Development 와 Test 환경도 최신 서버 코드를 사용할 수 있도록 같이 업데이트 필요	Make sure that you use a new queue and a new matchmaking configuration for your live environment. This will ensure that you are using server code that is up to date. If you have a development or test environment that is very old (anything over a year old qualifies), it may also be worthwhile recreating these too. If you have really old fleets, you can replace those too in the process. Now all the server code is the latest and greatest.
6	라이브 환경 서버에 대한 버전 업그레이드 테스트. 업그레이드 전략, 이슈 발생시 롤백 방법 검증. 빌드 업데이트를 고려하여 Fleet ID 대신 Alias 사용을 추천	Practice updates to the live environment servers. Be clear on your upgrade strategy, and how you will revert if you discover a problem with a new version. Consider using GameLift Aliases instead of fleet IDs in your system, (mandatory if not using GameLift game session queues).
7	라이브 환경 서버들이 문제없이 동작할 수 있는지 로드테스트로 점검	Validate with a load test that fleets in your live environment are available and performant.

8	라이브 fleet 들의 open 된 포트가 올바른지 점검. Public 하게 open 되어 있는지 (0.0.0.0/0) 그리고 실제 서버에서 오픈된 포트구성과 일치하는 지 재점검	Check that the ports open on live fleets match the range of ports that could possibly be used by any server on the fleet instances, and are open to 0.0.0.0/0. i.e. if servers listen to ports in the range 38200 to 38300, these must be open to the world.
9	인스턴스 타입과 서버당 process 갯수가 올바르게 설정되었는지 확인	Check that instance types and concurrent process limits are correct.
10	Live fleet 에 접근할 수 있는 IAM 권한이 올바르게 설정되어 있는지 확인	Validate that the correct group of people are IAM users of the Live fleets. Restrict access to other people by means of limiting Admin privileges and access to live resources.
11	RDP / SSH 포트가 막혀있거나 특정 필요 IP 에 한정해서 오픈되어있는지 점검. 불필요한 포트는 Any 로 Open 되어있지 않아야 함.	Check that RDP port 3389 and SSH port 22 are closed, or limited to a small address range reflecting the developer's public IP range(s). These ports (and any unused ports) should not be open to 0.0.0.0/0 under any circumstances.
12	Fleet 의 스케일링 정책에서 스케일 인 이벤트시에도 필요한 최소 수량을 확보할 수 있도록 0 보다 큰 값으로 최소 수량을 설정	Set fleet minimum sizes above zero, so that fleets can't accidentally turn their own scaling rules off during a scale-down event. Having a larger minimum gives small populations more flexibility, and these can be reduced later.
13	Fleet 최대 사이즈가 피크 로드를 수용할 수 있는지 검증. 예측하지 못한 트래픽을 대응하기 위해 일부 여유를 포함하는 것을 권장	Set fleet maximum sizes sufficiently high to accommodate at least peak anticipated demand, including a good margin of 2X or so for unanticipated demand.
14	Fleet 보호 정책을 Full protection 정책으로 설정하여, 활성화된 게임 세션이 있는 상태에서 해당 서버에 스케일 다운 이벤트가 발생하지 않도록 함	Set fleet protection policy to Full Protection on all live fleets so that scaling down cannot evict an active game session.

런치 당일

	확인 항목	원문
1	오픈 시 갑자기 들어오는 트래픽을 받을 수 있도록 미리 Fleet 내 서버 수량을 미리 확보해 둘 필요 있음. Fleet 의 스케일링 정책에서 일시적으로 minimum instance 수량을 조절하여 최소 서버 수량이 확보될 수 있도록 설정하는 방법도 가능	As you are ready for players to start joining, pre-warm the fleets by scaling them up to a level that can accommodate the launch spike. The scaling rules will try to bring them down if you leave it long enough, so keep an eye on this. Disabling scaling policy in the console for warming may be warranted.
2	관련 인원이 모두 war room 에 모여 상세 모니터링을 수행	Monitor everything very carefully as first players arrive. Encourage the whole team to participate in the event in the war room. Celebrate victory.
3	매칭 메이킹 (Flex Match 포함)의 경우 플레이어 수가 충분하지 않을 경우나 글로벌 플레이어 분포로 인해 예상대로 동작하지 않을 수도 있으므로 매칭 시간이 적절한지 모니터링 필요	Matchmakers, including Flexmatch, tend to do less well with small player pools and their behavior changes subtly with scale and global player distribution, so monitor particularly whether matching times are acceptable.
4	만족스러운 사용자 경험을 제공할 수 있도록 플레이어 Latency 에 대한 모니터링이 필요	Monitor that player latency is within tolerable limits, and players are getting a good experience.

런치 이후

	확인 항목	원문

1	플레이어의 요청을 받아들이는데 문제는 없으면서도 Idle capaicty 를 최소화 하기 위해 스케일링 정책에 대한 조율 필요.	Tune scaling rules so that you are able to minimize idle capacity, whilst at the same time supplying all players with games.
2	플레이어의 Latency 를 확인하여 매칭 규칙을 조절하거나 추가 Region 에 대한 배포를 고려	Measure the latency for your players, and modify matching rules or roll-out in additional regions based on the worst cases first.
3	사용자 피드백에 대한 확인. 변경에 대한 계획 수립	Listen to your players experiences, and plan changes with them in mind. Community managers are your vanguard.
4	서버 바이너리에 대한 최적화 고려. 인스턴스당 서버 프로세스 갯수를 늘릴 수 있다면 비용 효율성 증대	Optimize the server executable, as its performance efficiency has a direct bearing on the fleet costs. Increasing the number of server processes per instance is a good goal.
5	데이터 분석을 통해 개발, 사용자 경험, 지속성, 수익 창출에 최적화에 대한 고려. A/B 테스팅 진행	Use that analytics data to drive continued development, improving player experience, game longevity and optimize monetization (in that order), and also start A/B testing.